IDAMS

Internationally Developed Data Analysis and Management Software Package

WinIDAMS Reference Manual

(release 1.3)

April 2008

Copyright © 2001-2008 by UNESCO

Published by
the United Nations Educational, Scientific and Cultural Organization
Place de Fontenoy, 75700 Paris, France

© UNESCO ninth edition 2008

First published 1988

Revised 1990, 1992, 1993, 1996, 2001, 2003, 2004

Printed in France

UNESCO ISBN 92-3-102577-5 WinIDAMS Reference Manual

Preface

Objectives of IDAMS

The idea behind IDAMS is to provide UNESCO Member States free-of-charge with a reasonably comprehensive data management and statistical analysis software package. IDAMS, used in combination with CDS/ISIS (the UNESCO software for database management and information retrieval), will equip them with integrated software allowing for the processing in a unified way of both textual and numerical data gathered for scientific and administrative purposes by universities, research institutes, national administrations, etc. The ultimate objective is to assist UNESCO Member States to progress in the rationalization of the management of their various sectors of activity, a target which is crucial both to establish sound plans of development and for the monitoring of their execution.

Origin and a Short History of IDAMS

IDAMS was originally derived from the software package OSIRIS III.2 developed in the early seventies at the Institute for Social Research of the University of Michigan, U.S.A. It has been and is continuously enriched, modified and updated by the UNESCO Secretariat with the co-operation of experts from different countries, namely American, Belgian, British, Colombian, French, Hungarian, Polish, Russian, Slovak and Ukrainian specialists, hence the name of the software: "Internationally Developed Data Analysis and Management Software Package".

In the beginning there was IDAMS for IBM mainframe computers

The first release (1.2) was issued in 1988; it contained already almost all data management and most of the data analysis facilities. Although basic routines and a number of programs were taken from OSIRIS III.2, they were substantially modified and new programs were added providing tools for partial order scoring, factor analysis, rank-ordering of alternatives and typology with ascending classification. Features for handling code labels and for documenting program execution were incorporated. The software was accompanied by the User Manual, Sample Printouts and Quick Reference Card.

Release 2.0 was issued in 1990; in addition to regrouping of: (1) programs for calculating Pearsonian correlations; and (2) programs for rank-ordering of alternatives, it contained technical improvements in a number of programs.

Release 3.0 was issued in 1992; it contained significant improvements such as: harmonization of parameters, keywords and syntax of control statements, possibility of checking syntax of control statements without execution, possibility of program execution on limited number of cases, harmonization of error messages, possibility of aggregating and listing Recoded variables, alphabetic recoding and six new arithmetic functions in Recode facility. Two new programs were added: (1) for checking data consistency; and (2) for discriminant analysis. The Annex with statistical formulas was added to the User Manual.

Note: In 1993, after preparation of release 3.02 for both OS and VM/CMS operating systems, the development of the mainframe version was terminated.

In parallel, there was IDAMS for micro computers under MS-DOS

Development of micro computer version started in 1988 and was pursued in parallel with the development of the mainframe version until release 3.

The first release (1.0) was issued in 1989, with the same features and programs as the mainframe version.

Release 2.0 was issued in 1990; it was also fully compatible with the mainframe version. Moreover, the User Interface provided facilities for dictionary preparation, data entry, preparation and execution of setup files and printing of results.

Release 3.0 was issued in 1992 together with the mainframe version. However, the User Interface was made much more user friendly, providing new dictionary and data editors, a direct access to prototype setups for all programs as well as a module for interactive graphical exploration of data.

The two intermediate releases 3.02 and 3.04, issued in 1993 and 1994 respectively, included mainly internal technical improvements and debugging of a number of programs. Release 3.02 was the last one fully compatible with the mainframe version.

Micro IDAMS started its independent existence in 1993. The software underwent full and systematic testing, especially in the area of handling user errors, and it was fully debugged.

Release 4 (last release for DOS), issued in 1996, includes improved user-friendly interface, possibility of environment customization, on-line User Manual, simplified control language, new graphic presentation modalities and capability of producing national language versions. Two new programs came to give users cluster analysis and searching for structure techniques. The User Manual has been restructured in order to present topics in an easy-to-follow but concise way. It was available in English first.

Since 1998, the release 4 has been gradually developed in French, Spanish, Arabic and Russian.

2000: first version of IDAMS for Windows and further development

The release 1.0 of IDAMS for 32-bit Windows graphical operating system was given for testing in the year 2000 and its distribution started in 2001. It offers a modern user interface with a host of new features to improve ease-of-use and on-line access to the Reference Manual using standard Windows Help. New interactive components for data analysis provide tools construction of multidimensional tables, graphical exploration of data and time series analysis.

The release 1.1 was issued in September 2002 with the following improvements: (1) externalization of text that gives the possibility to have IDAMS software in other languages than English; (2) harmonization of text in the results. It was the first release of the Windows version which appeared in English, French and Spanish.

The release 1.2 was issued in July 2004 in English, French and Spanish with new functions in three programs, in the User Interface and in the interactive modules for graphical exploration of data and for time series analysis. It was issued in April 2006 in Portuguese.

The release 1.3 is also issued in English, French, Portuguese and Spanish, and contains new program for multivariate analysis of variance (MANOVA), calculation of coefficient of variation in four programs, improved handling of Recoded variables with decimals in SCAT and TABLES, and full harmonization of data record length.

Acknowledgements

First of all, thanks should go to Prof. Frank-M. Andrews († 1994) from the Institute for Social Research, University of Michigan, USA, as well as to the Institute who authorized UNESCO to take the OSIRIS III.2 source code and use it as a starting point in developing the IDAMS software package. Major improvements and additions have taken place since then. In this respect, particular gratitude should go to: Dr Jean-Paul Aimetti, Administrator of the D.H.E. Conseil, Paris and Professor at Conservatoire National des Arts et Métiers (CNAM), Paris (France); Prof. J.-P. Benzécri and E.-R. Iagolnitzer, U.E.R. de Mathématiques, Université de Paris V (France); Eng. Tibor Diamant and Dr Zoltán Vas, József Attila University, Szeged (Hungary); Prof. Anne-Marie Dussaix, Ecole Supérieure des Sciences Economiques et Commerciales (ESSEC), Cergy-Pontoise (France); Dr Igor S. Enyukov and Eng. Nicolaï D. Vylegjanin, StatPoint, Moscow (Russian Federation); Dr Péter Hunya, who has been the Director of the Kalmár Laboratory of Cybernetics, József Attila University, Szeged (Hungary), and IDAMS Programme Manager at UNESCO between July 1993 and February 2001; Jean Massol, EOLE, Paris (France); Prof. Anne Morin, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes (France); Judith Rattenbury, ex-Director, Data

Processing Division, World Fertility Survey, London, and presently founder and head of SJ MUSIC publishing house, Cambridge (United Kingdom); J.M. Romeder and Association pour le Développement et la Diffusion de l'Analyse des Données (ADDAD), Paris (France); Prof. Peter J. Rousseeuw, Universitaire Instelling Antwerpen, (Belgium); Dr A.V. Skofenko, Academy of Sciences, Kiev (Ukraine); Eng. Neal Van Eck, Susquehanna University, Selinsgrove (USA); Nicole Visart who has launched the IDAMS Programme at UNESCO and who, in addition to her technical contributions at all stages, assured the coordination and monitoring of the whole project until her retirement in 1992.

It is impossible to give due credit to all the many people, besides those already mentioned above, who have contributed ideas and effort to IDAMS and to OSIRIS III.2 from which it was derived. Up to now IDAMS has been developed mainly at UNESCO. Follows a list of names of the main programs, components and facilities included in WinIDAMS, with the names of authors and programmers, and the names of institutions where the work was done.

User Interface and Basic Facilities

Recode facility	Ellen Grun	ISR
	Peter Solenberger	ISR
User Interface	Jean-Claude Dauphin	UNESCO
On-line access to	Pawel Hoser	Polish Academy of Sciences
the Reference Manual	Jean-Claude Dauphin	UNESCO

Data Management Facilities

AGGREG	Tina Bixby	ISR
	Jean-Claude Dauphin	UNESCO
BUILD	Carl Bixby	ISR
	Sylvia Barge	ISR
	Tibor Diamant	UNESCO
CHECK	Tina Bixby	ISR
	Jean-Claude Dauphin	UNESCO
CONCHECK	Neal Van Eck	Van Eck Computing Consulting
CORRECT	Tibor Diamant	UNESCO
IMPEX	Péter Hunya	UNESCO
LIST	Marianne Stover	ISR
	Sylvia Barge	ISR
	Jean-Claude Dauphin	UNESCO
MERCHECK	Karen Jensen	ISR
	Sylvia Barge	ISR
	Zoltán Vas	JATE
MERGE	Tina Bixby	ISR
	Nancy Barkman	ISR
	Jean-Claude Dauphin	UNESCO
SORMER	Carol Cassidy	ISR
	Jean-Claude Dauphin	UNESCO
SUBSET	Judy Mattson	ISR
	Judith Rattenbury	ISR
	Jean-Claude Dauphin	UNESCO
TRANS	Jean-Claude Dauphin	UNESCO

Data Analysis Facilities

CLUSFIND	Leonard Kaufman	Vrije Universiteit Brussel
	Peter J. Rousseeuw	Vrije Universiteit Brussel
	Neal Van Eck	Van Eck Computing Consulting
	Tibor Diamant	UNESCO
CONFIG	Herbert Weisberg	ISR
DISCRAN	J.-M. Romeder and ADDAD	ADDAD
	Péter Hunya	UNESCO
	Tibor Diamand	UNESCO
FACTOR	J.P. Benzécri,	Université de Paris V
	E.R. Iagolnitzer	Université de Paris V
	Péter Hunya	JATE
MANOVA	Charles E. Hall	George Washington University
	Elliot M. Cramer	George Washington University
	Neal Van Eck	ISR
	Tibor Diamand	UNESCO
MCA	Edwin Dean	ISR
	John Sonquist	ISR
	Tibor Diamant	UNESCO
MDSCAL	Joseph Kruskal	Bell Telephone
	Frank Carmone	Bell Telephone
	Lutz Erbring	ISR
ONEWAY	Spyros Magliveras	ISR
	Tibor Diamant	UNESCO
PEARSON	John Sonquist	ISR
	Spyros Magliveras	ISR
	Neal Van Eck	ISR
	Ronald Nuttal	Boston College
	Tibor Diamant	UNESCO
POSCOR	Péter Hunya	JATE
QUANTILE	Robert Messenger	ISR
	Tibor Diamant	UNESCO
RANK	Anne-Marie Dussaix	ESSEC
	Albert David	ESSEC
	Péter Hunya	JATE
	A.V. Skofenko	Ukrainian Academy of Sciences
REGRESSN	M.A. Efroymson	ESSO Corporation
	Bob Hsieh	ESSO Corporation
	Neal Van Eck	ISR
	Peter Solenberger	ISR
SCAT	Judith Goldberg	ISR
SEARCH	John Sonquist	ISR
	Elizabeth Lauch Baker	ISR
	James N. Morgan	ISR
	Neal Van Eck	Van Eck Computing Consulting
	Tibor Diamant	UNESCO
TABLES	Neal Van Eck	ISR and Van Eck Computing Consulting
	Tibor Diamant	UNESCO
TYPOL	Jean-Paul Aimetti	CFRO
	Jean Massol	CFRO
	Péter Hunya	JATE
	Jean-Claude Dauphin	UNESCO
Multidimensional Tables	Jean-Claude Dauphin	UNESCO
GraphID	Igor S. Enyukov	StatPoint
	Nicolaï D. Vylegjanin	StatPoint
TimeSID	Igor S. Enyukov	StatPoint

As for the documentation, recognition should be expressed to all the people who contributed to its preparation, in particular to: Judith Rattenbury who drafted the first original English version of the Manual (1988) and who kept revising further editions till 1998; Jean-Paule Griset (UNESCO, Paris) who designed together with Nicole Visart the typography of the Manual used until 1998; Teresa Krukowska (IDAMS Group, UNESCO, Paris) who compiled the part with statistical formulas, changed the Manual's typography in 1998, continues updating the original English version since 1999, who is responsible for production of the Manual in English, French, Portuguese and Spanish, and takes care of harmonization, as much as possible, of texts in English, French, Portuguese and Spanish.

Acknowledgement to the authors of OSIRIS documents from which material was taken for WinIDAMS Reference Manual must be made as follows: the OSIRIS III.2 User Manual Vol.1 (edited by Sylvia Barge and Gregory A. Marks) and Vol.5 (compiled by Laura Klem), Institute for Social Research, University of Michigan, USA.

Thanks should also go to translators of the software and documentation into French, Portuguese and Spanish for their co-operation:

- Profesor José Raimundo Carvalho, CAEN Pós-graduação em Economia, UFC, Fortaleza, Brazil, for the translation of the Manual and texts as part of the software into Portuguese.
- Professor Bernardo Liévano, Escuela Colombiana de Ingeniería (ECI) Bogota, Colombia, for the translation of the Manual and texts as part of the software into Spanish.
- Professor Anne Morin, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes, France, for contribution to the translation into French of texts as part of the software.
- Nicole Visart, Grez-Doiceau, Belgium, for the translation of the Manual into French.

The following institutions have undertaken translation of the software and the Manual into Arabic and Russian: ALECSO - Department of Documentation and Information, Tunis, Tunisia, and Russian State Hydrometeorological University, Department of Telecommunications, St. Petersburg, Russian Federation.

Requests for WinIDAMS and Further Information

For further information on WinIDAMS regarding content, updating, training and distribution, please write to:

UNESCO
Communication and Information Sector
Information Society Division
CI/INF - IDAMS
1, rue Miollis
75732 PARIS CEDEX 15
France
e-mail: idams@unesco.org
http://www.unesco.org/idams

Contents

1 Introduction **1**
1.1 WinIDAMS User Interface . . . 1
1.2 Data Management Facilities . . . 2
1.3 Data Analysis Facilities . . . 2
1.4 Data in IDAMS . . . 4
1.5 IDAMS Commands and the "Setup" File . . . 5
1.6 Standard IDAMS Features . . . 5
1.7 Import and Export of Data . . . 6
1.8 Exchange of Data Between CDS/ISIS and IDAMS . . . 6
1.9 Structure of this Manual . . . 6

I Fundamentals **9**

2 Data in IDAMS **11**
2.1 The IDAMS Dataset . . . 11
2.1.1 General Description . . . 11
2.1.2 Method of Storage and Access . . . 11
2.2 Data Files . . . 11
2.2.1 The Data Array . . . 11
2.2.2 Characteristics of the Data File . . . 12
2.2.3 Hierarchical Files . . . 12
2.2.4 Variables . . . 12
2.2.5 Missing Data Codes . . . 13
2.2.6 Non-numeric or Blank Values in Numeric Variables - Bad Data . . . 13
2.2.7 Editing Rules for Variables Output by IDAMS Programs . . . 13
2.3 The IDAMS Dictionary . . . 14
2.3.1 General Description . . . 14
2.3.2 Example of a Dictionary . . . 16
2.4 IDAMS Matrices . . . 16
2.4.1 The IDAMS Square Matrix . . . 16
2.4.2 The IDAMS Rectangular Matrix . . . 18
2.5 Use of Data from Other Packages . . . 19
2.5.1 Raw Data . . . 19
2.5.2 Matrices . . . 19

3 The IDAMS Setup File **21**
3.1 Contents and Purpose . . . 21
3.2 IDAMS Commands . . . 21
3.3 File Specifications . . . 23
3.4 Examples of Use of $ Commands and File Specifications . . . 23
3.5 Program Control Statements . . . 24
3.5.1 General Description . . . 24
3.5.2 General Coding Rules . . . 25
3.5.3 Filters . . . 25
3.5.4 Labels . . . 26
3.5.5 Parameters . . . 27
3.6 Recode Statements . . . 31

4 Recode Facility **33**
4.1 Rules for Coding . . . 33
4.2 Sample Set of Recode Statements . . . 33
4.3 Missing Data Handling . . . 34
4.4 How Recode Functions . . . 34
4.5 Basic Operands . . . 35
4.6 Basic Operators . . . 35
4.7 Expressions . . . 36
4.8 Arithmetic Functions . . . 36
4.9 Logical Functions . . . 44
4.10 Assignment Statements . . . 45
4.11 Special Assignment Statements . . . 46
4.12 Control Statements . . . 47
4.13 Conditional Statements . . . 49
4.14 Initialization/Definition Statements . . . 50
4.15 Examples of Use of Recode Statements . . . 51
4.16 Restrictions . . . 54
4.17 Note . . . 55

5 Data Management and Analysis **57**
5.1 Data Validation with IDAMS . . . 57
5.1.1 Overview . . . 57
5.1.2 Checking Data Completeness . . . 57
5.1.3 Checking for Non-numeric and Invalid Variable Values . . . 58
5.1.4 Consistency Checking . . . 59
5.2 Data Management/Transformation . . . 59
5.3 Data Analysis . . . 60
5.4 Example of a Small Task to be Performed with IDAMS . . . 60

II Working with WinIDAMS **63**

6 Installation **65**
6.1 System Requirements . . . 65
6.2 Installation Procedure . . . 65
6.3 Testing the Installation . . . 65
6.4 Folders and Files Created During Installation . . . 66
6.4.1 WinIDAMS Folders . . . 66
6.4.2 Files Installed . . . 66
6.5 Uninstallation . . . 67

7 Getting Started **69**
7.1 Overview of Steps to be Performed with WinIDAMS . . . 69
7.2 Create an Application Environment . . . 70
7.3 Prepare the Dictionary . . . 71
7.4 Enter Data . . . 73
7.5 Prepare the Setup . . . 75
7.6 Execute the Setup . . . 76
7.7 Review Results and Modify the Setup . . . 76
7.8 Print the Results . . . 78

8 Files and Folders **79**
8.1 Files in WinIDAMS . . . 79
8.2 Folders in WinIDAMS . . . 80

9 User Interface **81**
9.1 General Concept . . . 81
9.2 Menus Common to All WinIDAMS Windows . . . 82
9.3 Customization of the Environment for an Application . . . 83
9.4 Creating/Updating/Displaying Dictionary Files . . . 85

9.5 Creating/Updating/Displaying Data Files . . . 86
9.6 Importing Data Files . . . 89
9.7 Exporting IDAMS Data Files . . . 90
9.8 Creating/Updating/Displaying Setup Files . . . 90
9.9 Executing IDAMS Setups . . . 92
9.10 Handling Results Files . . . 92
9.11 Creating/Updating Text and RTF Format Files . . . 93

III Data Management Facilities 95

10 Aggregating Data (AGGREG) 97
10.1 General Description . . . 97
10.2 Standard IDAMS Features . . . 97
10.3 Results . . . 98
10.4 Output Dataset . . . 98
10.5 Input Dataset . . . 99
10.6 Setup Structure . . . 100
10.7 Program Control Statements . . . 100
10.8 Restrictions . . . 102
10.9 Example . . . 102

11 Building an IDAMS Dataset (BUILD) 103
11.1 General Description . . . 103
11.2 Standard IDAMS Features . . . 104
11.3 Results . . . 104
11.4 Output Dataset . . . 105
11.5 Input Dictionary . . . 105
11.6 Input Data . . . 105
11.7 Setup Structure . . . 106
11.8 Program Control Statements . . . 106
11.9 Examples . . . 107

12 Checking of Codes (CHECK) 109
12.1 General Description . . . 109
12.2 Standard IDAMS Features . . . 109
12.3 Results . . . 109
12.4 Input Dataset . . . 110
12.5 Setup Structure . . . 110
12.6 Program Control Statements . . . 110
12.7 Restrictions . . . 112
12.8 Examples . . . 112

13 Checking of Consistency (CONCHECK) 115
13.1 General Description . . . 115
13.2 Standard IDAMS Features . . . 115
13.3 Results . . . 115
13.4 Input Dataset . . . 116
13.5 Setup Structure . . . 116
13.6 Program Control Statements . . . 116
13.7 Restrictions . . . 118
13.8 Examples . . . 118

14 Checking the Merging of Records (MERCHECK) 119
14.1 General Description . . . 119
14.2 Standard IDAMS Features . . . 120
14.3 Results . . . 121
14.4 Output Data . . . 121
14.5 Input Data . . . 121
14.6 Setup Structure . . . 122

14.7 Program Control Statements . . . 122
14.8 Restrictions . . . 124
14.9 Examples . . . 125

15 Correcting Data (CORRECT) 127
15.1 General Description . . . 127
15.2 Standard IDAMS Features . . . 127
15.3 Results . . . 128
15.4 Output Dataset . . . 128
15.5 Input Dataset . . . 128
15.6 Setup Structure . . . 128
15.7 Program Control Statements . . . 129
15.8 Restriction . . . 130
15.9 Example . . . 130

16 Importing/Exporting Data (IMPEX) 133
16.1 General Description . . . 133
16.2 Standard IDAMS Features . . . 133
16.3 Results . . . 133
16.4 Output Files . . . 134
16.5 Input Files . . . 135
16.6 Setup Structure . . . 137
16.7 Program Control Statements . . . 137
16.8 Restrictions . . . 139
16.9 Examples . . . 140

17 Listing Datasets (LIST) 143
17.1 General Description . . . 143
17.2 Standard IDAMS Features . . . 143
17.3 Results . . . 143
17.4 Input Dataset . . . 144
17.5 Setup Structure . . . 144
17.6 Program Control Statements . . . 144
17.7 Restriction . . . 145
17.8 Examples . . . 146

18 Merging Datasets (MERGE) 147
18.1 General Description . . . 147
18.2 Standard IDAMS Features . . . 147
18.3 Results . . . 148
18.4 Output Dataset . . . 148
18.5 Input Datasets . . . 149
18.6 Setup Structure . . . 150
18.7 Program Control Statements . . . 150
18.8 Restrictions . . . 153
18.9 Examples . . . 153

19 Sorting and Merging Files (SORMER) 155
19.1 General Description . . . 155
19.2 Standard IDAMS Features . . . 155
19.3 Results . . . 155
19.4 Output Dictionary . . . 155
19.5 Output Data . . . 155
19.6 Input Dictionary . . . 156
19.7 Input Data . . . 156
19.8 Setup Structure . . . 156
19.9 Program Control Statements . . . 157
19.10Restrictions . . . 157
19.11Examples . . . 158

20 Subsetting Datasets (SUBSET) 159
20.1 General Description . . . 159
20.2 Standard IDAMS Features . . . 159
20.3 Results . . . 159
20.4 Output Dataset . . . 160
20.5 Input Dataset . . . 160
20.6 Setup Structure . . . 160
20.7 Program Control Statements . . . 161
20.8 Restrictions . . . 162
20.9 Examples . . . 162

21 Transforming Data (TRANS) 163
21.1 General Description . . . 163
21.2 Standard IDAMS Features . . . 163
21.3 Results . . . 163
21.4 Output Dataset . . . 163
21.5 Input Dataset . . . 164
21.6 Setup Structure . . . 164
21.7 Program Control Statements . . . 165
21.8 Restrictions . . . 166
21.9 Examples . . . 166

IV Data Analysis Facilities 169

22 Cluster Analysis (CLUSFIND) 171
22.1 General Description . . . 171
22.2 Standard IDAMS Features . . . 171
22.3 Results . . . 171
22.4 Input Dataset . . . 172
22.5 Input Matrix . . . 172
22.6 Setup Structure . . . 173
22.7 Program Control Statements . . . 173
22.8 Restrictions . . . 175
22.9 Examples . . . 175

23 Configuration Analysis (CONFIG) 177
23.1 General Description . . . 177
23.2 Standard IDAMS Features . . . 177
23.3 Results . . . 177
23.4 Output Configuration Matrix . . . 178
23.5 Output Distance Matrix . . . 178
23.6 Input Configuration Matrix . . . 178
23.7 Setup Structure . . . 179
23.8 Program Control Statements . . . 179
23.9 Restrictions . . . 180
23.10Examples . . . 181

24 Discriminant Analysis (DISCRAN) 183
24.1 General Description . . . 183
24.2 Standard IDAMS Features . . . 183
24.3 Results . . . 183
24.4 Output Dataset . . . 184
24.5 Input Dataset . . . 185
24.6 Setup Structure . . . 185
24.7 Program Control Statements . . . 185
24.8 Restrictions . . . 188
24.9 Examples . . . 188

25 Distribution and Lorenz Functions (QUANTILE) 189

25.1 General Description . . . 189
25.2 Standard IDAMS Features . . . 189
25.3 Results . . . 189
25.4 Input Dataset . . . 190
25.5 Setup Structure . . . 190
25.6 Program Control Statements . . . 190
25.7 Restrictions . . . 192
25.8 Example . . . 192

26 Factor Analysis (FACTOR) 193
26.1 General Description . . . 193
26.2 Standard IDAMS Features . . . 193
26.3 Results . . . 194
26.4 Output Dataset(s) . . . 194
26.5 Input Dataset . . . 195
26.6 Setup Structure . . . 195
26.7 Program Control Statements . . . 196
26.8 Restrictions . . . 199
26.9 Examples . . . 199

27 Linear Regression (REGRESSN) 201
27.1 General Description . . . 201
27.2 Standard IDAMS Features . . . 202
27.3 Results . . . 203
27.4 Output Correlation Matrix . . . 203
27.5 Output Residuals Dataset(s) . . . 204
27.6 Input Dataset . . . 204
27.7 Input Correlation Matrix . . . 204
27.8 Setup Structure . . . 205
27.9 Program Control Statements . . . 205
27.10Restrictions . . . 208
27.11Examples . . . 208

28 Multidimensional Scaling (MDSCAL) 211
28.1 General Description . . . 211
28.2 Standard IDAMS Features . . . 212
28.3 Results . . . 212
28.4 Output Configuration Matrix . . . 213
28.5 Input Data Matrix . . . 213
28.6 Input Weight Matrix . . . 213
28.7 Input Configuration Matrix . . . 214
28.8 Setup Structure . . . 214
28.9 Program Control Statements . . . 214
28.10Restrictions . . . 216
28.11Example . . . 216

29 Multiple Classification Analysis (MCA) 217
29.1 General Description . . . 217
29.2 Standard IDAMS Features . . . 218
29.3 Results . . . 218
29.4 Output Residuals Dataset(s) . . . 219
29.5 Input Dataset . . . 220
29.6 Setup Structure . . . 220
29.7 Program Control Statements . . . 221
29.8 Restrictions . . . 222
29.9 Examples . . . 223

30 Multivariate Analysis of Variance (MANOVA) 225
30.1 General Description . . . 225
30.2 Standard IDAMS Features . . . 226

30.3 Results . . . 226
30.4 Input Dataset . . . 227
30.5 Setup Structure . . . 227
30.6 Program Control Statements . . . 228
30.7 Restrictions . . . 229
30.8 Examples . . . 229

31 One-Way Analysis of Variance (ONEWAY) 231
31.1 General Description . . . 231
31.2 Standard IDAMS Features . . . 231
31.3 Results . . . 231
31.4 Input Dataset . . . 232
31.5 Setup Structure . . . 232
31.6 Program Control Statements . . . 233
31.7 Restrictions . . . 234
31.8 Examples . . . 234

32 Partial Order Scoring (POSCOR) 235
32.1 General Description . . . 235
32.2 Standard IDAMS Features . . . 235
32.3 Results . . . 235
32.4 Output Dataset . . . 236
32.5 Input Dataset . . . 236
32.6 Setup Structure . . . 237
32.7 Program Control Statements . . . 237
32.8 Restrictions . . . 240
32.9 Examples . . . 240

33 Pearsonian Correlation (PEARSON) 243
33.1 General Description . . . 243
33.2 Standard IDAMS Features . . . 243
33.3 Results . . . 244
33.4 Output Matrices . . . 244
33.5 Input Dataset . . . 245
33.6 Setup Structure . . . 245
33.7 Program Control Statements . . . 245
33.8 Restrictions . . . 247
33.9 Examples . . . 247

34 Rank-Ordering of Alternatives (RANK) 249
34.1 General Description . . . 249
34.2 Standard IDAMS features . . . 250
34.3 Results . . . 250
34.4 Input Dataset . . . 251
34.5 Setup Structure . . . 252
34.6 Program Control Statements . . . 253
34.7 Restrictions . . . 254
34.8 Examples . . . 254

35 Scatter Diagrams (SCAT) 257
35.1 General Description . . . 257
35.2 Standard IDAMS Features . . . 257
35.3 Results . . . 258
35.4 Input Dataset . . . 258
35.5 Setup Structure . . . 258
35.6 Program Control Statements . . . 259
35.7 Restrictions . . . 260
35.8 Example . . . 260

36 Searching for Structure (SEARCH) 261

36.1 General Description . . . 261
36.2 Standard IDAMS Features . . . 261
36.3 Results . . . 262
36.4 Output Residuals Dataset . . . 262
36.5 Input Dataset . . . 263
36.6 Setup Structure . . . 263
36.7 Program Control Statements . . . 263
36.8 Restrictions . . . 266
36.9 Examples . . . 266

37 Univariate and Bivariate Tables (TABLES) 269
37.1 General Description . . . 269
37.2 Standard IDAMS Features . . . 270
37.3 Results . . . 270
37.4 Output Univariate/Bivariate Tables . . . 272
37.5 Output Bivariate Statistics Matrices . . . 272
37.6 Input Dataset . . . 272
37.7 Setup Structure . . . 273
37.8 Program Control Statements . . . 273
37.9 Restrictions . . . 278
37.10Example . . . 278

38 Typology and Ascending Classification (TYPOL) 281
38.1 General Description . . . 281
38.2 Standard IDAMS Features . . . 281
38.3 Results . . . 282
38.4 Output Dataset . . . 283
38.5 Output Configuration Matrix . . . 283
38.6 Input Dataset . . . 283
38.7 Input Configuration Matrix . . . 284
38.8 Setup Structure . . . 284
38.9 Program Control Statements . . . 284
38.10Restrictions . . . 287
38.11Examples . . . 287

V Interactive Data Analysis 289

39 Multidimensional Tables and their Graphical Presentation 291
39.1 Overview . . . 291
39.2 Preparation of Analysis . . . 291
39.3 Multidimensional Tables Window . . . 293
39.4 Graphical Presentation of Univariate/Bivariate Tables . . . 294
39.5 How to Make a Multidimensional Table . . . 294
39.6 How to Change a Multidimensional Table . . . 297

40 Graphical Exploration of Data (GraphID) 301
40.1 Overview . . . 301
40.2 Preparation of Analysis . . . 301
40.3 GraphID Main Window for Analysis of a Dataset . . . 301
40.3.1 Menu bar and Toolbar . . . 302
40.3.2 Manipulation of the Matrix of Scatter Plots . . . 304
40.3.3 Histograms and Densities . . . 305
40.3.4 Regression Lines (Smoothed lines) . . . 306
40.3.5 Box and Whisker Plots . . . 307
40.3.6 Grouped Plot . . . 307
40.3.7 Three-dimensional Scatter Diagrams and their Rotation . . . 308
40.4 GraphID Window for Analysis of a Matrix . . . 308
40.4.1 Menu bar and Toolbar . . . 309
40.4.2 Manipulation of the Displayed Matrix . . . 310

41 Time Series Analysis (TimeSID) 311
41.1 Overview . . . 311
41.2 Preparation of Analysis . . . 311
41.3 TimeSID Main Window . . . 311
41.3.1 Menu bar and Toolbar . . . 312
41.3.2 The Time Series Window . . . 313
41.4 Transformation of Time Series . . . 314
41.5 Analysis of Time Series . . . 315

VI Statistical Formulas and Bibliographical References 317

42 Cluster Analysis 319
42.1 Univariate Statistics . . . 319
42.2 Standardized Measurements . . . 319
42.3 Dissimilarity Matrix Computed From an IDAMS Dataset . . . 320
42.4 Dissimilarity Matrix Computed From a Similarity Matrix . . . 320
42.5 Dissimilarity Matrix Computed From a Correlation Matrix . . . 320
42.6 Partitioning Around Medoids (PAM) . . . 320
42.7 Clustering LARge Applications (CLARA) . . . 322
42.8 Fuzzy Analysis (FANNY) . . . 322
42.9 AGglomerative NESting (AGNES) . . . 323
42.10DIvisive ANAlysis (DIANA) . . . 324
42.11MONothetic Analysis (MONA) . . . 324
42.12References . . . 325

43 Configuration Analysis 327
43.1 Centered Configuration . . . 327
43.2 Normalized Configuration . . . 327
43.3 Solution with Principal Axes . . . 327
43.4 Matrix of Scalar Products . . . 328
43.5 Matrix of Interpoint Distances . . . 328
43.6 Rotated Configuration . . . 328
43.7 Translated Configuration . . . 328
43.8 Varimax Rotation . . . 328
43.9 Sorted Configuration . . . 329
43.10References . . . 329

44 Discriminant Analysis 331
44.1 Univariate Statistics . . . 331
44.2 Linear Discrimination Between 2 Groups . . . 331
44.3 Linear Discrimination Between More Than 2 Groups . . . 333
44.4 References . . . 334

45 Distribution and Lorenz Functions 335
45.1 Formula for Break Points . . . 335
45.2 Distribution Function Break Points . . . 335
45.3 Lorenz Function Break Points . . . 336
45.4 Lorenz Curve . . . 336
45.5 The Gini Coefficient . . . 336
45.6 Kolmogorov-Smirnov D Statistic . . . 336
45.7 Note on Weights . . . 337

46 Factor Analyses 339
46.1 Univariate Statistics . . . 339
46.2 Input Data . . . 340
46.3 Core Matrices (Matrices of Relations) . . . 340
46.4 Trace . . . 341
46.5 Eigenvalues and Eigenvectors . . . 341
46.6 Table of Eigenvalues . . . 341

46.7 Table of Principal Variables' Factors . . . 342
46.8 Table of Supplementary Variables' Factors . . . 343
46.9 Table of Principal Cases' Factors . . . 344
46.10Table of Supplementary Cases' Factors . . . 346
46.11Rotated Factors . . . 346
46.12References . . . 346

47 Linear Regression **347**
47.1 Univariate Statistics . . . 347
47.2 Matrix of Total Sums of Squares and Cross-products . . . 347
47.3 Matrix of Residual Sums of Squares and Cross-products . . . 348
47.4 Total Correlation Matrix . . . 348
47.5 Partial Correlation Matrix . . . 348
47.6 Inverse Matrix . . . 348
47.7 Analysis Summary Statistics . . . 349
47.8 Analysis Statistics for Predictors . . . 350
47.9 Residuals . . . 351
47.10Note on Stepwise Regression . . . 351
47.11Note on Descending Regression . . . 352
47.12Note on Regression with Zero Intercept . . . 352

48 Multidimensional Scaling **353**
48.1 Order of Computations . . . 353
48.2 Initial Configuration . . . 353
48.3 Centering and Normalization of the Configuration . . . 353
48.4 History of Computation . . . 354
48.5 Stress for Final Configuration . . . 356
48.6 Final Configuration . . . 356
48.7 Sorted Configuration . . . 356
48.8 Summary . . . 356
48.9 Note on Ties in the Input Data . . . 357
48.10Note on Weights . . . 357
48.11References . . . 358

49 Multiple Classification Analysis **359**
49.1 Dependent Variable Statistics . . . 359
49.2 Predictor Statistics for Multiple Classification Analysis . . . 360
49.3 Analysis Statistics for Multiple Classification Analysis . . . 361
49.4 Summary Statistics of Residuals . . . 362
49.5 Predictor Category Statistics for One-Way Analysis of Variance . . . 362
49.6 One-Way Analysis of Variance Statistics . . . 363
49.7 References . . . 363

50 Multivariate Analysis of Variance **365**
50.1 General Statistics . . . 365
50.2 Calculations for One Test in a Multivariate Analysis . . . 367
50.3 Univariate Analysis . . . 370
50.4 Covariance Analysis . . . 370

51 One-Way Analysis of Variance **371**
51.1 Descriptive Statistics for Categories of the Control Variable . . . 371
51.2 Analysis of Variance Statistics . . . 372

52 Partial Order Scoring **373**
52.1 Special Terminology and Definitions . . . 373
52.2 Calculation of Scores . . . 374
52.3 References . . . 375

53 Pearsonian Correlation **377**
53.1 Paired Statistics . . . 377
53.2 Unpaired Means and Standard Deviations . . . 378

53.3 Regression Equation for Raw Scores . . . 378
53.4 Correlation Matrix . . . 378
53.5 Cross-products Matrix . . . 378
53.6 Covariance Matrix . . . 378

54 Rank-ordering of Alternatives 379
54.1 Handling of Input Data . . . 379
54.2 Method of Classical Logic Ranking . . . 380
54.3 Methods of Fuzzy Logic Ranking: the Input Relation . . . 382
54.4 Fuzzy Method-1: Non-dominated Layers . . . 384
54.5 Fuzzy Method-2: Ranks . . . 385
54.6 References . . . 386

55 Scatter Diagrams 387
55.1 Univariate Statistics . . . 387
55.2 Paired Univariate Statistics . . . 387
55.3 Bivariate Statistics . . . 388

56 Searching for Structure 389
56.1 Means analysis . . . 389
56.2 Regression Analysis . . . 391
56.3 Chi-square Analysis . . . 392
56.4 References . . . 393

57 Univariate and Bivariate Tables 395
57.1 Univariate Statistics . . . 395
57.2 Bivariate Statistics . . . 396
57.3 Note on Weights . . . 402

58 Typology and Ascending Classification 403
58.1 Types of Variables Used . . . 403
58.2 Case Profile . . . 403
58.3 Group profile . . . 404
58.4 Distances Used . . . 404
58.5 Building of an Initial Typology . . . 405
58.6 Characteristics of Distances by Groups . . . 406
58.7 Summary Statistics for Quantitative Variables and for Qualitative Active Variables . . . 407
58.8 Description of Resulting Typology . . . 407
58.9 Summary of the Amount of Variance Explained by the Typology . . . 408
58.10Hierarchical Ascending Classification . . . 408
58.11References . . . 409

Appendix: Error Messages From IDAMS Programs 411

Index 413

Chapter 1

Introduction

IDAMS is a software package for the validation, manipulation and statistical analysis of data. It is organized as a collection of data management and analysis facilities accessible through a user interface and a common control language. Examples of the types of data that can be processed with IDAMS are: the answers to questions by respondents in a survey, information about books in a library, the personal characteristics and performance of students at a college, measurements from a scientific experiment. The common features of such data are that they consist of values of variables for each of a collection of objects/cases (e.g. in a sample survey, the questions correspond to the variables and the respondents to the cases).

Many different packages and programs exist for aid in the statistical analysis of such data. One special feature of IDAMS is that it also provides facilities for extensive data validation (e.g. code checking and consistency checking) before embarking on analysis. As far as analysis is concerned, IDAMS performs classical techniques such as table building, regression analysis, one-way analysis of variance, discriminant and cluster analysis and also some more advanced techniques such as principal components factor analysis and analysis of correspondences, partial order scoring, rank ordering of alternatives, segmentation and iterative typology. In addition, WinIDAMS provides for interactive construction of multidimensional tables, interactive graphical exploration of data and interactive time series analysis.

1.1 WinIDAMS User Interface

It is a multiple document interface (MDI) which allows to work simultaneously with different types of documents in separate windows.

The Interface provides the following:

- definition of Data, Work and Temporary folders for an application;
- Dictionary window for creating/updating/displaying Dictionary files;
- Data window for creating/updating/displaying Data files;
- Setup window to prepare/display Setup files;
- Results window to display, copy and print selected parts of results;
- general text editor;
- an option for executing IDAMS setups from a file or from the active Setup window;
- interactive data import/export facilities;
- access to interactive data analysis components (Multidimensional Tables, GraphID, TimeSID);
- on-line access to the Reference Manual.

1.2 Data Management Facilities

Aggregating data (AGGREG). Allows the grouping of records from a number of cases into one record and to output a new dataset with one record for each group, for example, records representing members of a household are grouped into household representing record. The variables in the new records are summary statistics of specified variables from the individual records, e.g. the sum, mean, minimum/maximum value.

Building an IDAMS dataset (BUILD). A raw data file (which may contain multiple records per case) is input along with a dictionary describing the variables to be selected. BUILD checks for non-numeric values in numeric fields; blank fields can be recoded to user-specified numeric values and other non-numerics are reported and replaced by 9's. The output is an IDAMS dataset comprising a Data file with a single record per case and a dictionary which describes each field in the data records.

Checking of codes (CHECK). Reports cases which have invalid variable values. Valid codes for each variable are specified by the user and/or taken from the dictionary.

Checking of consistency (CONCHECK). Reports cases with inconsistencies between two or more variables. IDAMS Recode statements are used to specify the logical relationships to be checked.

Checking the merging of records (MERCHECK). Checks that the correct records are present for each case in a file with multiple records per case. It outputs a file containing equal numbers of records per case. Invalid or duplicate records can be deleted and missing records can be inserted with missing values specified by the user.

Correcting data (CORRECT). Updates a Data file by applying corrections to individual variable values for specified cases. The Results file contains a written trace of corrections allowing them to be archived.

Importing/exporting data (IMPEX). Import is aimed at building IDAMS datasets or matrices from files coming from other software. The aim of export is to make possible the use of Data and Matrix files, stored in or created by IDAMS, in other packages. Free and DIF format text files can be imported/exported.

Listing datasets (LIST). Values for selected variables (original or recoded) and/or selected cases can be listed in the column format.

Merging datasets (MERGE). Two datasets can be merged by matching cases according to a common set of variables called match variables. There are 4 options for selecting cases for the output dataset: (1) only cases present in both files (intersection); (2) cases present in either file (union); (3) each case in the first file; (4) each case in the second file. The user specifies which variables from each of the two input files are to be output. An option exists for matching a case from one file with more than one case from the second file, e.g. for adding household data from one file to each individual's record in a second file.

Sorting and merging files (SORMER). This is a general purpose utility for sorting data into ascending or descending order on up to 12 fields. Up to 16 files may be merged.

Subsetting datasets (SUBSET). Outputs a new dataset (Data and Dictionary files) containing selected cases and/or variables from the input dataset. There is an option to check for duplicate cases.

Transforming data (TRANS). Allows variables created with the IDAMS Recode facility to be saved in a permanent dataset.

1.3 Data Analysis Facilities

Cluster analysis (CLUSFIND). Performs cluster analysis by partitioning a set of objects (cases or variables) into a set of clusters as determined by one of 6 algorithms, 2 based on partitioning around medoids, one based on fuzzy clustering and the other 3 based on hierarchical clustering.

Configuration analysis (CONFIG). Performs analysis on a single input configuration, created for example by MDSCAL program. It has the capability of centering, norming, rotating, translating dimensions, computing inter-point distances and scalar products. The configuration can be plotted after each transformation.

Discriminant analysis (DISCRAN). Looks for the best linear discriminant function(s) of a set of variables which reproduces, as far as possible, an a priori grouping of the cases. It uses a stepwise procedure, i.e. in each step the most powerful variable is entered. Three samples of cases can be distinguished: basic

sample on which the main discriminant analysis steps are performed, test sample on which the power of the discriminant function is checked and anonymous sample which is used only for classifying the cases. Case assignment and values of the two first discriminant factors (if there are more than 2 groups) can be saved in a dataset.

Distribution and Lorenz functions (QUANTILE). Distribution functions with 2 to 100 subintervals, Lorenz functions, Lorenz curve and Gini coefficients, and the Kolmogorov-Smirnov test.

Factor analysis (FACTOR). Covers a set of principal component factor analyses (scalar products, covariances, correlations) and factor analysis of correspondences. For each analysis, it constructs a matrix representing the relations between variables and computes its eigenvalues and eigenvectors. Then it calculates the case and/or variable factors giving for each case and/or variable its ordinate, its quality of representation and its contributions to the factors. Factors can be saved in a dataset and a graphic representation of cases and/or variables in the factor space can be obtained. Active and passive variables and cases can be distinguished.

Linear regression (REGRESSN). Multiple linear regression analysis: standard and stepwise. Either a dataset or a correlation matrix may be used as input. Residuals can be printed with the Durbin-Watson statistic for their first-order autocorrelation, and they can also be output for further analyses.

Multidimensional scaling (MDSCAL). This is a non-metric multidimensional scaling procedure for the analysis of similarities. Operates on a matrix of similarity or dissimilarity measures and looks for the best geometric representation of the data in n-dimensional space. The user controls the dimensionality of the configuration obtained, the distance metric used and the way the ties (equal values) in the input data should be handled.

Multiple classification analysis (MCA). Examines the relationships between several predictors and a single dependent variable, and determines the effect of each predictor before and after adjustment for its inter-correlations with other predictors. Provides information about bivariate and multivariate relationships between predictors and the dependent variable. Residuals can be printed and/or saved in a dataset.

Multivariate analysis of variance (MANOVA). Performs univariate and multivariate analysis of variance and of covariance, using a general linear model. Up to eight factors (independent variables) can be used. If more than one dependent variable is specified, both univariate and multivariate analyses are performed. The program performs an exact solution with either equal or unequal numbers of cases in the cells.

One-way analysis of variance (ONEWAY). Descriptive statistics of the dependent variable within categories of the control variable and one-way analysis statistics such as: total sum of squares, between means sum of squares, within groups sum of squares, eta and eta squared (unadjusted and adjusted) and the F-test value.

Partial order scoring (POSCOR). Calculates ordinal scale scores from interval or ordinal scale variables. Scores are calculated for each case involved in analysis and they measure the relative position of the case within the set of cases. The scores, optionally with other user-specified variables, are output in the form of an IDAMS dataset.

Pearsonian correlation (PEARSON). Calculates Pearson's r correlation coefficients, covariances, and regression coefficients. Pairwise or casewise deletion of missing data can be requested. Output correlation and covariance matrices can be saved in a file.

Rank-ordering of alternatives (RANK). Determines a reasonable rank-order of alternatives using preference data and three different ranking procedures, one based on classical logic and two others based on fuzzy logic. Preference data can represent either a selection or ranking of alternatives. Two types of individual preference relations can be specified: weak and strict. With fuzzy ranking, the data completely determine the results obtained whereas with classical ranking the user has the possibility of controlling the calculations.

Scatter diagrams (SCAT). Scatter diagrams, univariate statistics (mean, standard deviation and N) and bivariate statistics (Pearson's r and regression statistics: coefficient B and constant A).

Searching for structure (SEARCH). A binary segmentation procedure to develop predictive models. The question "what dichotomous split on which predictor variable will give the maximum improvement in the ability to predict values of the dependent variable" embedded in an iterative scheme, is the basis of the algorithm used.

Univariate and bivariate tables (TABLES). Options include: (1) univariate simple and cumulative

frequency and percentage distributions; (2) univariate statistics: mean, median, mode, variance, standard deviation, skewness, kurtosis, minimum, maximum; (3) bivariate frequency tables with row, column and total percentages; (4) tables of mean values of an additional variable; (5) bivariate statistics: t-test of means between pairs of rows, Chi-square, contingency coefficient, Cramer's V, Kendall's Taus, Gamma, Lambdas, Spearman rho, a number of statistics for Evidence Based Medicine, and 3 non-parametric tests: Wilcoxon, Mann-Whitney and Fisher.

Typology and ascending classification (TYPOL). Creates a typology variable as a summary of a large number of variables both quantitative and qualitative. The user chooses the initial and final number of groups, the type of distance used, and the way the initial typology is started. The groups of initial typology are stabilized using an iterative procedure. The number of groups can be reduced using an algorithm of hierarchical ascending classification. A distinction can be made between active variables which participate in the construction of typology, and passive variables, for which main statistics are calculated within the groups of the typology.

Interactive multidimensional tables. This component allows to visualize and customize multidimensional tables with frequencies, row, column and total percentages, summary statistics (sum, count, mean, maximum, minimum, variance, standard deviation) of additional variables, and bivariate statistics. Up to seven variables can be nested in rows or in columns. Construction of a table can be repeated for each value of up to three "page" variables. The tables can also be printed, or exported in free format (comma or tabulation character delimited) or in HTML format.

Interactive graphical exploration of data. A separate component, GraphID, is available for exploring data through graphic displays. The basic display is in the form of multiple scatterplots for different pairs of variables. Additional information such as histograms and regression lines may be displayed on each plot. The plots may be manipulated in various ways. For example, selected cases can be marked in one plot and then highlighted in all the other plots. Parts of the display may be enlarged ("zoomed"). IDAMS matrices are displayed as three dimensional plots with rows and columns being represented by two of the axes and the third dimension being used to show the size of the statistic for each cell.

Interactive time series analysis. Another separate component, TimeSID, provides a possibility for interactive analysis of time series. It contains analysis of trends, auto-correlations and cross-correlations, statistical and graphical analysis of time series values, tests of randomness and trends, forecasting for short terms, periodograms and estimation of spectral densities. Series can be transformed by calculating averages, arithmetic compositions, sequential differences, rates of change, smoothed by moving averages and decomposed using frequency filters.

1.4 Data in IDAMS

IDAMS dataset - the Data file. The data file input to IDAMS may be any character (ASCII) fixed format file, i.e. the values for a given variable occupy the same position (field) in the record for every case. Characteristics of this file are:

- 1-50 records per case;
- each case can contain up to 4096 characters;
- number of cases limited by the disk capacity and the internal representation of numbers;
- variables can be numeric (up to 9 characters) or alphabetic (up to 255 characters).

IDAMS dataset - the Dictionary file. The dictionary is used to describe the data:

- it may contain up to 1000 variables identified by a unique number between 1 and 9999;
- for each variable, it contains at minimum the variable's number, its type (numeric or alphabetic), and its location in the data record;
- for each variable, a variable name, two missing data codes, the number of decimal places and a reference number may also be specified;

- for qualitative variables, codes and corresponding labels may be included.

The pair of files consisting of a Dictionary file and the Data file it describes is known as an **IDAMS dataset.**

IDAMS matrices. Some analysis programs use a square or rectangular matrix as input rather than the raw data.

The ***square matrix*** is used for symmetric arrays of bivariate statistics with a constant on the diagonal. Only the upper right-hand corner of the matrix is stored, without the diagonal.

The ***rectangular matrix*** is for non-symmetric arrays of values. The meaning of the rows and columns varies according to the IDAMS program.

1.5 IDAMS Commands and the "Setup" File

With the exception of WinIDAMS interactive components, execution of an IDAMS program is launched by a setup. The setup contains information such as file specifications, program control statements, variable recoding instructions, etc., separated by IDAMS commands (starting with a $ character) which identify the kind of information being specified. The first IDAMS command in the Setup file always identifies the first program to be executed, e.g.

```
$RUN TABLES
$FILES
DICTIN = name of Dictionary file
DATAIN = name of Data file
$SETUP
    control statements for TABLES program
$RECODE
    variable recoding statements
```

1.6 Standard IDAMS Features

Case selection. By default all cases from a Data file will be processed in a program execution. To select a subset, a filter statement is included in the setup, e.g. INCLUDE V3=1 (include only those cases where variable 3 is equal to 1).

Variable selection. Variables are referenced by their numbers assigned in the dictionary. A set of variables is specified in a variable list following keywords such as VARS, CONVARS, OUTVARS. Such variable lists may also include R-variables constructed by the IDAMS Recode facility (see below), e.g. VARS=(V3-V6,V129,R100,R101).

Transforming/recoding data. A powerful Recode facility permits the recoding of variables and the construction of new variables. Recoding instructions are prepared by the user in the IDAMS Recode language. This includes the possibility of arithmetic computation as well as the use of several special functions for operations such as the grouping of values, the creation of "dummy" variables, etc. Conditional statements are also allowed. Examples of Recode statements for constructing 3 new variables R100, R101 and R102 are:

```
R100=V4+V5
R101=BRAC(V10,0-15=1,16-60=2,61-98=3,99=9)
IF (MDATA(V3,V4) OR V4 EQ 0) THEN V102=99 ELSE R102=V3*100/V4
```

The R-variables thus constructed for each case can be used temporarily in the program being executed or can be saved in a dataset using the TRANS program.

Weighting data. When complex sampling procedures are used during data collection, it may be necessary to use different weights for cases during analysis. Such weights are usually stored as a variable in the Data file. The WEIGHT parameter is then used in the program control statements to invoke weighting, e.g. WEIGHT=V5.

Treatment of missing data and "bad" data. Special values for each numeric variable can be identified as missing data codes and stored in the dictionary. During data processing missing data is handled through two parameters:

- MDVALUES (specifies which missing data codes are to be used to check for missing data in numeric variables);
- MDHANDLING (specifies what is to be done if missing data are encountered).

Normally it is assumed that data have been cleaned prior to analysis. If this is not the case, then the BADDATA parameter is available for skipping cases with non-numeric values (including blank fields) in numeric fields, or for treating such values as missing data.

1.7 Import and Export of Data

IDAMS does not use special internal file format for storing data. Any character file in fixed format can be described by an IDAMS dictionary and then input to IDAMS. On the other hand, free format data with Tab, comma or semicolon used as separator can be imported through the WinIDAMS User Interface. Moreover, the IMPEX program allows a fixed format IDAMS file to be created from any text file in free or DIF format.

Data files created by IDAMS are always character files in fixed format. Such files can be used directly by other software along with the appropriate data descriptive information for that software. Free format files with Tab, comma or semicolon used as separator can be obtained through the WinIDAMS User Interface. Moreover, the IMPEX program allows a fixed format IDAMS file to be exported as a text file in free or DIF format.

IDAMS matrices are stored in a format specific to IDAMS (described in the "Data in IDAMS" chapter). The IMPEX program can be used to import/export free format matrices.

1.8 Exchange of Data Between CDS/ISIS and IDAMS

There is a separate program, WinIDIS, which prepares data description and performs data transfer between IDAMS and CDS/ISIS (the UNESCO software for database management and information retrieval). Such transfer is controlled by IDAMS and ISIS data description files (the IDAMS dictionary and the CDS/ISIS Field Definition Table). When going from ISIS to IDAMS, a new IDAMS Dictionary and Data files are always constructed and they can be merged with other data using IDAMS data management facilities. When going from IDAMS to ISIS, there are three possibilities: (1) a completely new data base can be constructed, (2) transferred records can be added to an existing data base as new data base records, (3) records of an existing data base can be updated with the transferred data.

1.9 Structure of this Manual

All the general features of IDAMS, including the Recode facility, are described in ***Part 1*** of this Manual.

Part 2 includes installation instructions, description of files and folders used in WinIDAMS, a section entitled "Getting Started" which takes a user through the steps required to perform simple task, and description of the WinIDAMS User Interface.

In-depth descriptions of each IDAMS program are given in ***Parts 3 and 4***. These write-ups contains the following sections:

General Description. A statement of the primary purpose of the program.

Standard IDAMS Features. Statements about the case and variable selection possibilities, data transformation, weighting capabilities, and missing data handling.

Results. Details of results destined to be printed (or reviewed on the screen).

Description of output and input files. One section for each IDAMS dataset, each matrix and each other input or output file, giving a description of their contents.

Setup Structure. A designation of the file specifications, IDAMS commands, and program control statements needed to execute the program.

Program Control Statements. The parameters and/or formats of each of the program control statements with an example of each type.

Restrictions. A summary of the program limitations.

Examples. Examples of complete sets of control statements for executing the program.

Part 5 provides description of WinIDAMS interactive components for construction of multidimensional tables, for graphical exploration of data and for time series analysis.

Part 6 provides details of statistical techniques, formulas and bibliographical references for all analysis programs.

Finally, errors issued by IDAMS programs are summarized in the Appendix.

Part I

Fundamentals

Chapter 2

Data in IDAMS

2.1 The IDAMS Dataset

2.1.1 General Description

The dataset consists of 2 separate files: a Data file and a Dictionary file which describes some or all of the fields (variables) in the records of the data file. All Dictionary/Data files output by IDAMS programs are IDAMS datasets.

2.1.2 Method of Storage and Access

Both Dictionary and Data files are read and written sequentially. Thus they may be stored on any media. There is no special IDAMS internal "system" file as in some other packages. The files are in character/text format (ASCII) and can be processed at any time with general utilities or editors, or input directly to other statistical packages.

2.2 Data Files

2.2.1 The Data Array

Irrespective of its actual format in the data file, the data can be visualized as a rectangular array of variable values, where element x_{ij} is the value of the variable represented by the j-th column for the case represented by the i-th row. For example, the data from a survey can be displayed in the following way:

Cases	Variables					
	identification	education	sex	age	...	
case 1	1300	6	2	31	...	...
case 2	1301	2	1	25	...	
.	1302	3	1	55	...	
.	.	.	.	.	...	

In the example, each row represents a respondent in a survey and each column represents an item from the questionnaire.

2.2.2 Characteristics of the Data File

These files contain normally, but not necessarily, fixed length records, since the end of the record is recognized by carriage return/line feed characters. However, the length of the longest record must be supplied on the file definition (see $FILES command). There is no limit to the number of records in the Data file.

The maximum record length is 4096 characters.

Each "case" may consist of more than one record (up to a maximum of 50). If, in a particular program execution, variables are to be accessed from more than one type of record, then there must be exactly the same number of records for each case. The MERCHECK program can be used to create files complying with this condition. Note that any Data file output by an IDAMS program is always restructured to contain a single record per case.

If a raw data file contains different record types (and the record type is coded) and does not have exactly the same number of records per case, IDAMS programs can be executed using variables from one record type at a time by selecting only that record type at the start.

2.2.3 Hierarchical Files

IDAMS only processes "rectangular" files as described above. Hierarchical files can be handled by storing records from the different levels in different files and then using the AGGREG and MERGE programs to produce composite records containing variables from the different levels. Alternatively, the complete hierarchical data file can be processed one level at a time by "filtering" records for that level only (providing record types are coded).

2.2.4 Variables

Referencing variables. The variables in a Data file are identified by a unique number between 1 and 9999. This number, preceded by a V (e.g. V3) is used to refer to a particular variable in control statements to programs. The variable number is used to index a variable-descriptor record in the dictionary which provides all other necessary information about the variable such as its name and its location in the data record.

Variable types. Variables can be of numeric or alphabetic type, both stored in character mode.

Numeric variables. These can be positive or negative valued with the following characteristics:

- A value can be composed of the numeric characters 0-9, a decimal point and a sign (+,-). Leading blanks are allowed.
- Values must be right justified in the field (i.e. with no trailing blanks) unless an explicit decimal point appears.
- Maximum field width is 9 but only up to 7 significant digits (both integers and decimals taken together) are retained in processing.
- Variable values can be integers (e.g. an age variable or a categorical variable such as sex) or may be decimal valued (e.g. a variable with percentage values). The number of decimals (NDEC) is stored in the variable's descriptor record in the dictionary. Normally the decimal point is "implicit" and does not appear in the data. In this case NDEC gives the number of digits of the variable's value that are to be treated as decimal places. If an "explicit" decimal point is coded in the data, then NDEC is used to determine the number of digits to the right of the decimal point that will be retained, rounding up the value if necessary, e.g. values coded 4.54 and 4.55 with NDEC=1 will be used as 4.5 and 4.6 respectively.
- A sign (if it appears) must be the first character, e.g. "-0123".
- Blank fields are considered non-numeric and treated as "bad" data. See below for how to deal with blanks used in the data to indicate missing or inapplicable data.
- With the exception of BUILD, all IDAMS programs accept values in exponential notation, e.g. value coded .215E02 will be used as 21.5 .

Alphabetic variables. Alphabetic variables can be held in Data files and can be up to 255 characters long. They can be used in data management programs. 1-4 character alphabetic variables can be also used in filters. In order to be used in analysis, 1-4 character alphabetic variables must be recoded to numeric values. This can be done with Recode's BRAC function.

2.2.5 Missing Data Codes

The value of a variable for a particular case may be unknown for a number of reasons, for example a question may be inapplicable to certain respondents or a respondent may refuse to answer a question. Special missing data codes can be established for each numeric variable and coded into the data when needed. Two missing data codes are allowed: MD1 and MD2. If used, any value in the data equal to MD1 is considered a missing value; any value greater than or equal to MD2 (if MD2 is positive or zero) or less than or equal to MD2 (if MD2 is negative) is also considered missing.

These missing data codes are stored in the dictionary record for the variable. Similar to data values, they can be integer or decimal valued, with an implicit or explicit decimal point. If MD1 or MD2 is specified with an implicit decimal point, NDEC gives the number of digits to be treated as decimal places. If an explicit decimal point is coded in MD1 or MD2, then NDEC determines the number of digits to the right of the decimal point to be retained, rounding up the value accordingly.

When a variable's MD1 and MD2 codes are blank in the dictionary, this means that there are no special numeric missing data codes. During an IDAMS program execution, blank dictionary MD1 and MD2 fields are filled in by the default missing data codes of 1.5×10^9 and 1.6×10^9 respectively.

Since the missing data codes are each limited to a maximum of 7 digits (or 6 digits and a negative sign), they can present a problem for 8 and 9 digit variables. The user should consider the use of a negative first missing data code in this case.

2.2.6 Non-numeric or Blank Values in Numeric Variables - Bad Data

In IDAMS data management programs, data values are merely copied from one place to another and conversion to a computational (binary) mode is not carried out; in this case there is no check on whether numeric variables have numeric values. However, when variables are being used for analysis or in Recode operations, then their values are converted to binary mode and values containing non-numeric characters will cause problems. Normally data should be cleaned of such characters prior to analysis. In addition, blank values in numeric variables are not automatically treated as missing values; they are also considered to be non-numeric or "bad" data.

To allow for analysis of incompletely cleaned data and for the handling of unrecoded blank fields, the BADDATA parameter may be used to treat blank and other non-numeric values as missing and thus have the possibility of eliminating them from analysis. Specification of the parameter BADDATA=MD1 or BADDATA=MD2 results in the conversion of "bad" values to the MD1 or MD2 code for the variable. If the MD1 or MD2 codes are blank, then bad data values are converted to the corresponding default missing data code (see above) and are thus treated as missing values (see the description of BADDATA parameter in "The IDAMS Setup File" chapter).

2.2.7 Editing Rules for Variables Output by IDAMS Programs

IDAMS programs always create a Data file and a corresponding IDAMS dictionary, i.e. an IDAMS dataset.

The Data file contains one record for each case. The record length is the sum of the field widths of all variables output and is determined by the program.

Numeric variable values are edited to a standard form as described below:

- If the entire field contains only the numeric characters 0-9, these are output exactly as they appear in the input data.
- If the field contains a number entered with leading blanks (e.g. ' 5'), the blanks are converted to zeros before the data are output. Fields with trailing blanks (e.g. '04 ' in a three digit numeric field), embedded blanks (e.g. '0 4') and all blanks are treated according to the BADDATA specification.
- If the field contains a positive value or a negative value with the '+' and '-' characters explicitly entered, the positive sign is removed and the negative sign is put before the first significant numeric digit.
- If the field contains a number with an explicit decimal point, this is removed and the value output has the same width as the input field and n decimal places as defined in the NDEC field of the variable description. Leading blanks in the field are converted to zeros. If more than n digits are found in the input field after the decimal point, the value is rounded and output to n decimal places (e.g. if n=2, an input value of 2.146 will be output as 215; if n=0, an input value of 1.5 will be output as 002). Trailing blanks do not cause an error condition. If fewer than n digits are found, zeros are inserted on the right for the missing decimal places.
- Values which are too big to fit into the field assigned are treated according to BADDATA specification.

Alphabetic variable values are not edited and are the same on input and output.

2.3 The IDAMS Dictionary

2.3.1 General Description

The dictionary is used to describe the variables in the data. For each variable it must contain at minimum the variable's number, its type and its location in the data record. In addition, a variable name, two missing data codes, the number of decimal places and a reference number or name may be given. This information is stored in variable-descriptor records sometimes known as T-records. Optional C-records for categorical variables give labels for the different possible codes. The first record in the dictionary, the dictionary-descriptor record, identifies the dictionary type, gives the first and last variable numbers used in the dictionary and specifies the number of data records making up a "case".

The original dictionary is prepared by the user to describe the raw data. IDAMS programs which output datasets always produce new dictionaries reflecting the new format of the data.

Dictionary records have fixed format and are 80-characters long.

A detailed description of each type of dictionary record is given below.

Dictionary-descriptor record. This is always the first record in the dictionary.

Columns	*Content*
4	3 (indicates the type of dictionary).
5-8	First variable number (right justified).
9-12	Last variable number (right justified).
13-16	Number of records per case (right justified).
20	Form in which variable location is specified (columns 32-39) on the variable-descriptor records.
	blank — Record number and starting and ending columns. Record length must be 80 to use this format if the number of records per case is > 1.
	1 — Starting location and field width.

Variable-descriptor records (T-records). The dictionary contains one such record for each variable. These records are arranged in ascending order by the variable number. The variable numbers need not be contiguous. The maximum number of variables is 1000.

Columns	*Content*
1	T
2-5	Variable number.
7-30	Variable name.
32-39	Location; according to column 20 of the dictionary-descriptor record.
	Either
	32-33 Record sequence number containing starting column of variable.
	34-35 Starting column number.
	36-37 Record sequence number containing ending column of variable.
	38-39 Ending column number.
	Or
	32-35 Starting location of the variable within the case.
	36-39 Field width (1-9 for numeric variables and 1-255 for alphabetic variables).
40	Number of decimal places (numeric variables only). Blank implies no decimal places.
41	Type of variable.
	blank Numeric.
	1 Alphabetic.
45-51	First missing data code for numeric variables (or blanks if no 1st missing data code). Right justified.
52-58	Second missing data code for numeric variables (or blanks if no 2nd missing data code). Right justified.
59-62	Reference number (optional - can be used to contain some unchangeable alphanumeric reference for the variable, e.g. the original variable number or a question reference).
73-75	Study ID (optional - can be used to identify the study to which this dictionary belongs).

Note 1: When record and column numbers are used to indicate variable location, listings of the dictionary records do not show the record and column numbers as they appear on the dictionary record. Rather, the variable location is translated to and printed in the starting location/width format. For example, for a variable in columns 22-24 of the third record of a multiple record (record length 80) per case data file, the starting location will be 182 (2 * 80 + 22) and the width 3.

Note 2: If there is more than one record per case and the record length is not 80, then starting location and field width notation must be used on the T-records. The starting location is counted from the start of the first record. For example, for records of length 121, the starting location of a field at position 11 of the 2nd record for a case would be 132.

Code-label records (C-records). The dictionary may optionally contain these records for any of the variables. They follow immediately after the T-record for the variable to which they apply and provide codes and their labels for different possible values of the variable. They are used by programs such as TABLES to print row and column labels along with the corresponding codes. They can also be used as the specification of valid codes for a variable during data entry with the WinIDAMS User Interface and for data validation with the program CHECK.

Columns	*Content*
1	C
2-5	Variable number.
6-9	Reference number (optional - can be used to contain some unchangeable alphanumeric reference for the variable, e.g. the original variable number or a question reference).
15-19	Code value **left** justified.
22-72	Label for this code. (Note that only the first 8 characters will be used by analysis programs printing code labels although the complete label will appear in listings of the dictionary).
73-75	Study ID (optional).

2.3.2 Example of a Dictionary

```
Columns:          1         2         3         4         5         6...
         12345678901234567890123456789012345678901234567890123456789012345678901...

            3   1  20   1   1
         T   1 Identification              1   5
         T   2 Age                         6   2          99
         T   3 Sex                         8   1
         C   3         1      Female
         C   3         2      Male
         T  11 Region                     16   1
         C  11         1      North
         C  11         2      South
         C  11         3      East
         C  11         4      West
         T  12 Grade average              17   31        000    900
         T  20 Name                       31  30 1
```

This is a dictionary describing 6 data fields in a data record as shown diagrammatically below.

1-5	6-7	8		16	17-19		31-60	
V1 ID	V2 Age	V3 Sex		V11 Region	V12 Grade		V20 Name	

Locations of variables are expressed in terms of starting position and field width (1 in column 20 of dictionary-descriptor) and there is one record per case (1 in column 16). There is one implied decimal place in the grade average variable (V12). The age variable has a code 99 for missing data. For the grade average, 0's imply missing data as do all values greater than or equal to 90.0. The name of each respondent (V20) is recorded as a 30 character alphabetic (type 1) variable. Note that variable numbers need not be contiguous and that not all fields in the data need to be described.

2.4 IDAMS Matrices

There are two types of IDAMS matrices: square and rectangular. Both types are self-described, but unlike the IDAMS dataset, the "dictionary" is stored in the same file as the array of values. In general, these matrices are created by one IDAMS program to be used as input to another program and the user need not be familiar with the format. If, however, it is necessary to prepare a similarity matrix, a configuration matrix, etc. by hand, then the formats described below must be observed.

Regardless of type, all records are fixed length 80-character records.

2.4.1 The IDAMS Square Matrix

The square matrix can be used only for a square and symmetric array. Only the values in the upper-right triangular, off-diagonal portion of the array are actually stored in the square matrix. An array of Pearsonian correlation coefficients is suitably stored like this.

Programs which input/output square matrices. PEARSON outputs square matrices of correlations and covariances; REGRESSN outputs square matrix of correlations; TABLES outputs square matrices of bivariate measures of association. These matrices are appropriate input to other programs, e.g. the correlation matrix output from PEARSON can be input to REGRESSN and to CLUSFIND. Moreover, CLUSFIND and MDSCAL input square matrix of similarities or dissimilarities.

Example.

```
          Columns:                1111111111222222222 3...
                          12345678901234567890123456789 0...

Matrix descriptor           2   4
Format statements     |  #F  (12F6.3)
                      |  #F  (6E12.5)
Variable identifi-    |  #T   1 AGE
         cations      |  #T   3 EDUCATION
                      |  #T   9 RELIGION
                      |  #T  10 SEX
Array of values       |   -.011 -.174 -.033
                      |    .131 -.105
                      |   -.133
Means & standard      |   0.33350E 01 0.54950E 01 0.50251E 01 0.40960E 01
        deviations    |   0.20010E 01 0.19856E 01 0.15000E 01 0.12345E 01
```

Format. The square matrix contains the following:

1. A matrix-descriptor record. This, the first record, gives the matrix type and the dimensions of the array of values.

Columns	*Content*
4	2 (indicates **square** matrix).
5-8	The number of variables (right justified).

2. A Fortran format statement describing each row of the array of values. The format statement describes the number of value fields per 80-character record and the format of each. For example, a format of (12F6.3) indicates that each row of the array is recorded with up to 12 values per record, each value occupying 6 columns, 3 of which are decimals. If a row contains more than 12 values, a new record contains the 13th value, etc. Each new row of the array always starts on a new record.

Columns	*Content*
1-2	#F
3-80	The format statement, enclosed in parentheses.

3. A Fortran format statement describing the vectors of the variable means and standard deviations. The format statement describes the number of values per record and the format of each.

Columns	*Content*
1-2	#F
3-80	The format statement, enclosed in parentheses.

4. Variable identification records. These are n records, where n is the number of variables specified on the matrix-descriptor record. The order of these records corresponds to the order of variables indexing the rows (and columns) of the array of values. When a matrix is created by an IDAMS program, the variable numbers and names are retained from the IDAMS dataset from which the bivariate statistics were generated.

Columns	*Content*
1-2	#T or #R (indicates variable identification for a row of the matrix).
3-6	The variable number (right justified).
8-31	The variable name.

   The above four sections of the matrix are referred to as the matrix "dictionary". Following the matrix dictionary is the array of values.

5. The array of values. Since the array is symmetric and has diagonal cells usually containing a constant (e.g. a correlation of 1.0 for a variable correlated with itself), only the off-diagonal, upper-right corner of the array is stored. Note that for a covariance matrix the diagonal elements can be calculated using standard deviations which are included in the matrix file (see point 7 below).

   In the example of the 4-variable matrix above, the full array (before entering in the square format) would be as follows:

```
vars        1       3       9      10
 1      1.000   -.011   -.174   -.033
 3      -.011   1.000    .131   -.105
 9      -.174    .131   1.000   -.133
10      -.033   -.105   -.133   1.000
```

The portion of the array that is stored is:

```
vars        1       3       9      10
 1              -.011   -.174   -.033
 3                       .131   -.105
 9                              -.133
10
```

Each row of this reduced array begins a new record and is written according to the format specification in the matrix dictionary (see above).

6. A vector of variable means. The n values are recorded in accordance with the format statement in the matrix dictionary.
7. A vector of variable standard deviations. The n values are recorded in accordance with the format statement in the matrix dictionary.

2.4.2 The IDAMS Rectangular Matrix

The rectangular matrix differs from the square matrix in that the array of values may be square (and non-symmetric) or rectangular. Further, since the rows of some arrays are not indexed by variables, e.g. a frequency table, the rectangular matrix may or may not contain variable identification records; the rectangular matrix does not contain variable means and standard deviations.

Programs which input/output rectangular matrices. These matrices are created by the CONFIG, MDSCAL, TABLES and TYPOL programs. They are appropriate input for CONFIG, MDSCAL and TYPOL.

Example.

```
          Columns:                    11111111112222222223...
                                 12345678901234567890123456789 0...

Matrix descriptor                   3   4   3
Format statement                  #F (l6F5.0)
Variable identifications     | #T   2 IQ
                             | #T   5 EDUCATION
                             | #T   8 MOBILITY
                             | #T  12 SIBLING RIVALRY
Array of values              |    59   20   10
                             |    37   15    2
                             |    50   40    7
                             |     8   26   31
```

Format. The rectangular matrix contains the following:

1. A matrix-descriptor record.

Columns	*Content*
4	3 (indicates **rectangular** matrix).
5-8	The number of rows (right justified).
9-12	The number of columns (right justified).
16	Number of format (#F) statement records. (Blank implies 1).
20	Presence of row and column labels.
	blank/0 Row labels only are present (#R or #T records).

	1	Column labels only are present (#C records).
	2	Row and column labels are present (#R or #T, and #C records).
	3	No row or column labels are present.
21-40	Row variable name (optional).	
41-60	Column variable name (optional).	
61-80	Description of the matrix contents (optional):	
		Weighted frequencies
		Unweighted freqs
		Row percentages
		Column percentages
		Total percentages
		Name of the variable for which mean values are included in the matrix.

2. A Fortran format statement describing each row of the array of values. The format describes an 80-character record. For example, a format of (16F5.0) indicates that each row of the array is recorded with up to 16 values per record and with each value occupying 5 columns, none of which is a decimal place.

Columns	*Content*
1-2	#F
3-80	The format statement, enclosed in parentheses.

3. Variable identification records. The order of these records corresponds to the order of the variables/codes indexing the rows and columns of the matrix. When a rectangular matrix is created by an IDAMS program, the variable/code numbers and names are retained from the input dataset or matrix from which the array of values was derived.

Columns	*Content*
1-2	#T or #R for row labels, #C for column labels.
3-6	The variable number or the code value (right justified). The code values longer than 4 characters are replaced by ****.
8-58	The variable name or the code label.

   The above three sections of the matrix are referred to as the matrix "dictionary". Following the matrix dictionary is the array of values.

4. The array of values. The full array is stored. Each row of the array begins a new record and is written according to the format specified in the matrix dictionary.

2.5 Use of Data from Other Packages

2.5.1 Raw Data

Any data in the form of fixed format records in character (ASCII) mode can be input directly to IDAMS programs. Nearly all data base and statistical packages have an "export" or "convert" function to produce fixed format character mode data files. An IDAMS dictionary must be prepared to describe the fields required from the data.

Free format data files with Tab, comma or semicolon used as separator can be imported directly through the WinIDAMS User Interface. See the "User Interface" chapter for details.

Free format (any character being used as delimiter including blank) and DIF format text files can also be imported using the IMPEX program.

Data stored in an CDS/ISIS data base can be imported to IDAMS using the WinIDIS program.

2.5.2 Matrices

The IMPEX program can be used to import free format matrices. Furthermore, matrices produced outside IDAMS, for example a matrix provided in a publication, may also be entered according to the format given above.

Chapter 3

The IDAMS Setup File

3.1 Contents and Purpose

To execute IDAMS programs, the user prepares a special file called the "Setup" file which controls the execution of the programs. This file contains IDAMS commands and control statements necessary for execution such as: reference to program to be executed, the names of files, the options to be selected for the program and variable transformation instructions, e.g.

```
$RUN program name
$FILES
    file specifications
$SETUP
    program control statements
$RECODE
    Recode statements
```

3.2 IDAMS Commands

These commands, which start with a "$", separate the different kind of information being provided for an IDAMS program execution. Available commands are:

$RUN program	(name of program to be executed)
$FILES [RESET]	(signals start of file specifications)
$RECODE	(signals start of Recode statements)
$SETUP	(signals start of program control statements)
$DICT	(signals start of dictionary)
$DATA	(signals start of data)
$MATRIX	(signals start of a matrix)
$PRINT	(turns printing on and off)
$COMMENT [text]	(comments)
$CHECK [n]	(checking if previous step terminated well).

The first line in a Setup file must always be a $RUN command identifying the IDAMS program to be executed. Other commands relating to this program execution (followed by associated control statements or data) can be placed in any order. These are then followed by the $RUN command for the next program (if any) to be executed and so on. The individual IDAMS commands are described below in alphabetical order.

$CHECK [n]. If this command is present, the program will not be executed if the immediately preceding program terminated with a condition code greater than n. If the command is present, but no value is supplied, the value of n defaults to 1.

- All IDAMS programs terminate with a condition code of 16 if setup errors are encountered. For example, if TABLES is to be executed immediately after TRANS, but the user does not want to execute TABLES if a setup error occurred in the TRANS execution, a $CHECK command after the $RUN TABLES command will prevent execution of TABLES.
- The $CHECK command may appear anywhere in the setup for the program, but is usually placed immediately after the $RUN command.

$COMMENT [text]. The "text" from this command is printed in the listing of the setup. This command has no effect on program execution.

$DATA. The $DATA command signals that the data follow.

- This feature cannot be used if the program generates an output Data file and a DATAOUT file is not specified, i.e. the data are output to a default temporary file.
- This feature cannot be used if the $MATRIX feature is used.
- The record length of data in the setup cannot exceed 80 characters. If longer records or lines are input, only the first 80 characters will be used.
- The print switch is turned off by the $DATA command. Thus, unless a $PRINT command immediately follows the $DATA command, the data will not be printed.

$DICT. The $DICT command signals that an IDAMS dictionary follows.

- This feature cannot be used if the program generates an output dictionary and a DICTOUT file is not specified, i.e. if the dictionary is output to a default temporary file.
- The print switch is turned off by the $DICT command. Thus, unless a $PRINT command immediately follows the $DICT command, the dictionary will not be printed.

$FILES [RESET]. This signals the start of file specifications. Default file names are attached to each file at the start of IDAMS program(s) execution through the use of a special file "idams.def". Any of these default names may be changed by introducing file specification statements after the $FILES command (see "File Specifications" below). To get back default file names for Fortran FT files (except FT06 and FT50), use "FILES RESET" command.

$MATRIX. The $MATRIX command signals that a matrix or set of matrices follows.

- This feature cannot be used if the $DATA feature is used.
- The print switch is turned off by the $MATRIX command. Thus, unless a $PRINT command immediately follows the $MATRIX command, the matrix input will not be printed.

$PRINT. The print switch is reversed; if it was on, $PRINT will turn it off; if it was off, $PRINT will turn it on. When printing is on, lines from the Setup file are listed as part of the program results.

- When a $RUN command is encountered, the print switch is always turned on. The $DICT, $DATA, and $MATRIX commands automatically turn the print switch off.

$RECODE. The occurrence of this command signals that the IDAMS Recode facility is to be used. The Recode facility is described in the "Recode Facility" chapter of this manual.

- The Recode statements normally follow the $RECODE command. If a new IDAMS command follows immediately after a $RECODE command, Recode statements from the setup for the preceding program will be used.

$RUN program. $RUN specifies the program to be executed and always is the first statement in the setup.

- "program" is the 1 to 8 character name of the program.

- All commands and statements following the $RUN command and up to the next $RUN command apply to the program named.
- The print switch is turned on when $RUN is encountered. See the $PRINT description.

$SETUP. The $SETUP command signals the beginning of the program control statements, i.e. the filter, label, parameter statement, etc. (see below).

- The $SETUP command is required even when program control statements follow immediately after the $RUN command.

3.3 File Specifications

The names of the files to be used are given following the $FILES command and take the following format:

ddname=filename [RECL=maximum record length]

where:

- ddname is the file reference name used internally by programs, e.g. DICTIN. The required files and the corresponding ddnames for a particular program are given in the program write-up in the section "Setup Structure".
- filename is the physical file name. Enclose the name in primes if it contains blanks. See section "Folders in WinIDAMS" for additional explanation.
- RECL must be used if the first record in a Data file is not the longest. If RECL is not specified the record length is taken as the record length of the first record. If a subsequent record is longer, an input error results.

Examples:

```
DATAIN = A:ECON.DAT RECL=92
PRINT  = RSLTS.LST
FT02   = ECON.MAT
DICTIN = \\nec0102\commondata\econ.dic
```

For additional explanation, see section "Customization of the Environment for an Application" in the "User Interface" chapter.

3.4 Examples of Use of $ Commands and File Specifications

Example A. Perform multiple executions of an analysis program, e.g. ONEWAY using the same data but with, for instance, different filters.

```
$RUN ONEWAY
$FILES
DICTIN = CHEESE.DIC
DATAIN = CHEESE.DAT
$SETUP
     Filter 1
     Other control statements for ONEWAY
$RUN ONEWAY
$SETUP
     Filter 2
     Other control statements for ONEWAY
```

Example B. Execute TABLES and ONEWAY, using the same Dictionary and Data files for each and using the same Recode; do not list the Recode statements.

```
$RUN TABLES
$FILES
DICTIN = ABC.DIC
DATAIN = ABC.DAT  RECL=232
$SETUP
     Control statements for TABLES
$RECODE
$PRINT
     Recode statements
$RUN ONEWAY
$SETUP
     Control statements for ONEWAY
$RECODE
$COMMENT THE RECODE STATEMENTS INPUT FOR TABLES WILL BE REUSED FOR ONEWAY
```

Example C. Execute TABLES using IDAMS Recode, dictionary in the setup, data on diskette. Print the input dictionary.

```
$RUN TABLES
$FILES
DATAIN = A:MYDATA
$RECODE
     Recode statements
$SETUP
     Control statements for TABLES
$DICT
$PRINT
     Dictionary
```

Example D. Use the output from a data management program as input to analysis programs without retaining the output file, e.g. execute TRANS followed by TABLES using the output data from TRANS by specifying parameter INFILE=OUT. TABLES is not to be executed if the TRANS has control statement errors.

```
$RUN TRANS
$FILES
DICTIN = MYDIC4
DATAIN = MYDAT4
$SETUP
     Control statements for TRANS
$RECODE
     Recode statements
$RUN TABLES
$CHECK
$SETUP
     Control statements for TABLES including parameter INFILE=OUT
```

3.5 Program Control Statements

3.5.1 General Description

IDAMS program control statements (which follow the $SETUP command) are used to specify the parameters for a particular execution. There are three standard control statements used by all programs:

1. the optional filter statement for selecting the cases from the data file to be used,

2. the mandatory label statement which assigns a title for the execution,
3. a mandatory parameter statement which selects the options for the program; some program options are standard across most programs, others are program specific.

Additional program control statements required by individual programs are described in the program write-up.

3.5.2 General Coding Rules

- Control statements are entered on lines up to 255 characters long.
- Lines may be continued by entering a dash at the end of one line and continuing on the next.
- The maximum length of information that may be entered for one control statement is 1024 characters excluding the continuation characters.
- Lower case letters, except for those occurring in strings enclosed in primes, are converted to upper case.
- If character strings enclosed in primes are included on a control statement, these should be continued in one line.

3.5.3 Filters

Purpose. A filter statement is used to select a subset of data cases. It is expressed in terms of variables and the values assumed by those variables. For example, if variable V5 indicates "sex of respondent" in a survey and code 1 represents female, then "INCLUDE V5=1" is a filter statement which specifies female respondents as the desired subset of cases.

The main filter selects cases from an input Data file and applies throughout a program execution. These filters are available with all IDAMS programs which input a dictionary (except BUILD and SORMER). Some programs allow for additional subsetting. Such "local" filtering applies only to a specific program action, e.g. one frequency table.

Examples.

```
1. INCLUDE V2=1-5 AND V7=23,27,35 AND V8=1,2,3,6
2. EXCLUDE V10=2-3,6,8-9 AND V30=<5 OR V91=25
3. INCLUDE V50='FRAN','UK','MORO','INDI'
```

Placement. If a main filter is used, it is always the first program control statement. Each program write-up indicates whether "local" filters may also be used.

Rules for coding.

- The filter statement begins with the word INCLUDE or EXCLUDE. Depending on which word is given, the filter statement defines the subset of cases to be used by the program (INCLUDE) or the subset to be ignored (EXCLUDE).
- A statement may contain a maximum of 15 expressions. An expression consists of a variable number, an equals sign, and a list of possible values. The list of values can contain individual values and/or ranges of values separated by commas, e.g. V2=1,5-9. Open ended ranges are indicated by < or >, e.g. INCLUDE V1=0,3-5,>10; however the variable must always be followed by an = sign to begin with, e.g. V1>0 must be expressed V1=>0 and V1<0 as V1=<0.
- Expressions are connected by the conjunctions AND and OR.
  - AND indicates that a value from each of the series of expressions connected by AND must be found.
  - OR indicates that a value from at least one of a series of expressions connected by OR must be found.

- Expressions connected by AND are evaluated before expressions connected by OR. For example, "expression-1 OR expression-2 AND expression-3" is interpreted as "expression-1 OR (expression-2 AND expression-3)". Thus, in order for a case to be in the subset defined by these expressions, either a value from expression-1 occurs, values from both expression-2 and expression-3 occur, or a value from each of the three expressions occurs.
- Parentheses **cannot** be used in the filter statement to indicate precedence of expression evaluation.
- Variables may appear in any order and in more than one expression. However, note that "V1=1 OR V1=2" is equivalent to the single expression "V1=1,2". Note also that "V1=1 AND V1=2" is an impossible condition, as no single case can have both a '1' and a '2' as a value for variable V1.
- A filter statement may optionally be terminated by an asterisk.
- The variables in a filter.
  - Numeric and alphabetic character type variables can be used.
  - R-variables **are not** allowed in main filters. They **are** allowed in analysis specific or local filters. Note that the REJECT statement in Recode can be used to filter cases on R-variables.
- The values in a filter for numeric variables.
  - Numeric values may be integer or decimal, positive or negative, e.g. 1, 2.4, -10.
  - Values are expressed singly or in ranges and are separated by commas, e.g. 1-5, 8, 12-13.
  - For numeric filter variables, variable values in the data file are first converted to real binary mode using the correct number of decimal places from the dictionary and the comparison with the filter value is then done numerically. Note that this means that for a variable with decimals, filter values must be given with the decimal point in the correct place, e.g. V2=2.5-2.8.
  - Cases for which a filter variable has a non-numeric value are always excluded from the execution.
- The values in a filter for alphabetic variables.
  - Values of 1-4 characters are expressed as character strings enclosed in primes, e.g. 'F'. Blanks on the right need not be entered, i.e. trailing blanks will be added.
  - If the variable has a field width greater then 4, only the first 4 characters from the data are used for the comparison with the filter variable.
  - Only single values, separated by commas are allowed; ranges of character strings cannot be used.

Note. The first statement following a $SETUP command is recognized as a main filter if it starts with INCLUDE or EXCLUDE. If the first non-blank characters are anything else, the statement is assumed to be a label.

3.5.4 Labels

Purpose. A label statement is used to title the results of a program execution. Some IDAMS programs print this label once at the start of the results, while others use it to title each page.

Examples.

```
1. TABLES ON 1998 ELECTION DATA - JULY, 2000
2. PRINTING OF CORRECTED A34 SURVEY DATA
```

Placement. A label statement **is required** by all IDAMS programs. The label is either the first or (if a filter is used), the second program control statement. If no special labeling is desired, it is still necessary to include a blank line.

Rules for coding.

- The statement may be a string of any characters from which the first 80 characters are used, i.e. if a label longer than 80 characters is input, it is truncated to the first 80.
- If the label is not enclosed in primes, lower case letters are converted to upper case and blanks are reduced to one blank.
- The label should not begin with the words "INCLUDE" or "EXCLUDE".

3.5.5 Parameters

Purpose. All IDAMS programs have been designed in a fairly general way, allowing the user to select among several options. These options and values are generated by parameters and are supplied on program control statements, such as "parameters", "regression specifications", "table specifications", etc. Parameters are specified by the user in a standard keyword format with an English word or abbreviation being used to identify an option.

Examples.

```
1. WRITE=CORR WEIGHT=V3, PRINT=(DICT, PAIR)
   (PEARSON - parameters)
2. DEPV=V5 METHOD=STEP VARS=(R3-R9,V30) WRITE=RESID
   (REGRESSN - regression parameters)
3. ROWV=(V3,V9,V10) COLV=(V4,V11,V19) CELLS=(FREQ,ROWPCT) STATS=(CHI,TAUA)
   (TABLES - table description)
```

Placement. The main parameter statement **is required** by all IDAMS programs and it must follow the label statement. If all defaults are chosen, a line with a single asterisk must be supplied. Each program write-up indicates the type and content of any other parameter lists that are required and indicates their position relative to other program control statements.

Presentation of keyword parameters in the program write-ups. All write-ups have a standard notation in the sections which describe the program parameters which are available. The basic notation is as follows:

- A slash indicates that only one of the mutually exclusive items can be chosen, e.g. SAMPLE/POPUL or PRINT=CDICT/DICT.
- A comma indicates that all, some, or none of the items may be chosen, e.g. STATS=(TAUA, TAUB, GAMMA).
- When commas and slashes are combined, only one (or none) of the items from each group separated by commas and connected by slashes may be chosen, e.g. PRINT=(CDICT/DICT, LONG/SHORT).
- Defaults, if any, are in bold, e.g. METHOD=**STANDARD**/STEPWISE/DESCENDING. A default is a parameter setting that the program assumes if an explicit selection is not made by the user.
- When a parameter setting is obligatory but has no default, the words "No default" are used.
- Words in upper case are keywords. Words or phrases in lower case indicate that the user should replace the word or phrase with an appropriate value, e.g. MAXCASES=n, VARS=(variable list).

Types of keywords. There are 5 types of keywords used for specifying parameters.

1. A keyword followed by a character string. This type of keyword identifies a parameter consisting of a string of characters, e.g.

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input dictionary and data files.

A user might specify:

INFILE=IN2
(the ddnames would be DICTIN2 and DATAIN2)

2. A keyword followed by one or more variable numbers, e.g.

   WEIGHT=variable number
   The weight variable number if the data are to be weighted.

   VARS=(variable list)
   Use only the variables in the list; the numbers may be listed in any order with or without V-notation, i.e. VARS=(V1-V3) or VARS=(1-3). Note that the program write-ups always indicate whether V- and R-type variables or only V-type variables may be used.

   A user might specify:

   WEIGHT=V39
   (the weight variable is V39)

   VARS=(32,1,10)
   (only the variables specified are to be used)

3. A keyword followed by one or more numeric values, e.g.

   MAXCASES=n
   Only the first n cases will be processed.

   IDLOC=(s1,e1,s2,e2, ...)
   Starting and ending columns of 1-5 case identification fields.

   A user might specify:

   MAXCASES=100
   (only the first 100 cases will be used)

   IDLOC=(1,3,7,9)
   (case ID is located in columns 1-3 and 7-9)

4. A keyword followed by one or more keyword values. The keyword values may be a mixture of mutually exclusive options (separated by slashes) and independent options (separated by commas). For example:

   PRINT=(**OUTDICT**/OUTCDICT/NOOUTDICT,DATA)

OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
NOOU	Do not print output dictionary.
DATA	Print the values of the output variables.

   A user might specify:

   PRINT=(OUTC,DATA)
   (full output dictionary is printed, and data values are printed)

   PRINT=NOOUTDICT
   (no output dictionary or data values are printed)

5. A set of mutually exclusive keywords. Only one of a set of options can be selected, e.g.

   SAMPLE/POPULATION

SAMP	Compute the variance and/or standard deviation using the sample equation.
POPU	Use the population equation.

All keywords except the last type are followed by an equals sign. The character, numeric, and keyword values that follow the equals sign are called the "associated values".

Rules for coding.

Rules for specifying keywords

- Only the first four letters of a keyword or an associated keyword need to be specified, although the whole keyword may be supplied. Thus, "TRAN" is an appropriate abbreviated form of the keyword "TRANSVARS". There are no abbreviations for keywords with four letters or less.

Rules for specifying associated values

- Associated value is a list of items.
  - The items in the list are separated by commas.
  - If there are two or more items, the list must be enclosed in parentheses.
  - Ranges of integer numeric values or variables are indicated by a dash.
  - Ranges of decimal numeric values are not allowed.

  For example:

```
R=(V2,3,5)
PRIN=(DICT,DATA,STAT)
MAXC=5
TRAN=(V5,V10-V25,V32)
IDLOC=(1,3,7,8)
```

- Associated value is a character string.
  - The string must be enclosed in primes if it contains any non-alphanumeric characters, e.g. FNAME='EDUCATION: WAVE 1'. Note that blank, dot and comma are non-alphanumeric characters. When in doubt, use primes.
  - Two consecutive primes (not a quotation mark) must be used to represent a prime, e.g, ANAME='KEVIN"S' (the extra prime is deleted, once the string is read).
  - A string is better not split across lines.

Rules for specifying lists of keywords

- Keywords (with or without associated values) are separated from one another by a comma or by one or more blanks, e.g.

```
FNAME='FRED', TRAN=3    KAISER
```

- Lists of keywords may spread across several lines but in this case there must be a dash (-) at the end of each line indicating continuation, e.g.

```
FNAME='FRED' -
TRAN=3 -
KAISER
```

- Keywords may be given in any order. If a keyword appears more than once in the list, then the last value encountered is used.
- A keyword may not be split across lines.
- Each list of keywords may optionally be terminated by an asterisk.
- If all default options are chosen, a line with a single asterisk must be supplied.

Details of most common parameters not described fully in each program write-up.

1. **BADDATA.** Treatment of non-numeric data values.

   BADDATA=**STOP**/SKIP/MD1/MD2

   When non-numeric characters (including embedded blanks and all-blank fields) are found in numeric variables, the program should:

   STOP Terminate the execution.
   SKIP Skip the case.
   MD1 Replace non-numeric values by the first missing data code (or 1.5×10^9 if 1st missing data code is not specified).

MD2 Replace non-numeric values by the second missing data code (or 1.6×10^9 if 2nd missing data code is not specified).

For SKIP, MD1, and MD2 a message is printed about the number of cases so treated.

2. **MAXCASES.** The maximum number of cases to be processed.

   MAXCASES=n

   The value given is the maximum number of cases that will be processed. If n=0, no cases are read; this option can be used to test setups without reading the data. If the parameter is not specified at all, all cases from the input file are processed.

3. **MDVALUES.** Specify which, if either, of the missing data codes are to be used to check for missing data in variable values. Note that some programs have, in addition, a MDHANDLING parameter to specify how data values which are missing are to be handled.

   MDVALUES=**BOTH**/MD1/MD2/NONE

   BOTH Variable values will be checked against the MD1 codes and against the ranges of codes defined by MD2.
   MD1 Variable values will be checked only against the MD1 codes.
   MD2 Variable values will be checked only against the ranges of codes defined by MD2.
   NONE MD codes will not be used. All data values will be considered valid.

   The default is always that both MD codes are used.

4. **INFILE, OUTFILE.** Specifying ddnames with which input and output dictionary and data files are defined.

   INFILE=**IN**/xxxx
   OUTFILE=**OUT**/yyyy

   Input and output Dictionary and Data files for IDAMS programs are defined with ddnames DICTxxxx, DATAxxxx, DICTyyyy and DATAyyyy. These normally default to DICTIN, DATAIN, DICTOUT, DATAOUT. If several IDAMS programs are being executed in one setup, for example programs using different datasets as input, or when using the output from one program as input directly to another (chaining), then it is sometimes necessary to change these defaults.

5. **WEIGHT.** This parameter specifies the variable whose values are to be used for weighting data cases.

   WEIGHT=variable number

   The variable specified may be a V-type or R-type, integer or decimal valued. Cases with missing, zero, negative and non-numeric weight values are always skipped and a message is printed about the number of cases so treated. If the WEIGHT parameter is not specified, no weighting is performed.

6. **VARS.** This parameter and similar ones such as ROWVARS, OUTVARS, CONVARS, etc. are used to specify a list of variables.

   VARS=(variable list)

   If more than one variable is specified, the list must be enclosed in parentheses.

   Rules for specifying variable lists

   - Variables are specified by a variable "number" preceded by a V or an R. A V denotes a variable from an IDAMS dataset or matrix. An R denotes a resultant variable from a Recode operation. Note that internal to the programs and in the results, V- and R-type variables are distinguished by the sign of the variable number; positive numbers denote V-type variables and negative numbers denote R-type variables.
   - To specify a set of contiguously numbered variables, such as V3, V4, V5, V6, connect two variable numbers, each preceded by a V, with a dash (e.g. V3-V6 is valid; V3-6 is invalid). Use ranges with caution if the dataset has gaps in the variable numbering, as all variables within the range must appear in the dataset or matrix, i.e. V6-V8 implies V6,V7,V8. If V7 is not in the dictionary, then an error message will result. V-type and R-type variables may not be mixed in a range, i.e. V2-R5 is invalid.
   - Single variable numbers or ranges of variable numbers are separated by commas.
   - In general, for data management programs, variables may be listed more than once, while for analysis programs specifying a variable more than once is inappropriate and will cause termination. See the program write-up for details.

- Blanks may be inserted anywhere in the list.
- In general, variables may be specified in any order. The order of variables may, however, have special meaning in some programs; check the program write-up for details.

Examples:

```
VARS=(V1-V6, V9, V16, V20-V102, V18, V11, V209)
OUTVARS=(R104, V7, V10-V12, R100-R103,  -
     V16, V1)
CONVARS=V10
```

3.6 Recode Statements

The IDAMS Recode facility permits the temporary recoding of data during execution of IDAMS programs. Results from such recoding operations (together with variables transferred from the input file) can also be saved in permanent files using the TRANS program.

Recoding is invoked by the $RECODE command. This command and the associated Recode statements are placed after the $RUN command for the program with which the Recode facility is to be used. For example:

```
$RUN program                    $RUN ONEWAY
$FILES                          $FILES
File definitions                DICTIN=MYDIC
                                DATAIN=MYDAT
$RECODE                         $RECODE
Recode statements                R10 = BRAC(V3,0-10=1,11-20=2)
                                 R11 = SUM(V7,V8)
                                 NAME R10 'EDUC LEVEL', R11'TOTAL INCOME'
$SETUP                          $SETUP
Program control statements       INCOME BY EDUC,SEX
                                 BADDATA=SKIP
                                 CONVARS=(R10,V2) DEPVAR=R11
```

A complete description of the Recode facility is provided in the "Recode Facility" chapter.

Chapter 4

Recode Facility

4.1 Rules for Coding

- Recode statements take the form:

  lab statement

  where lab is an optional 1-4 character label starting in position 1 of the line and followed by at least one blank. Unlabelled statements must start in position 2 or beyond.

- The label allows control statements such as GO TO to refer to a specific statement, e.g. GO TO ST1. Labels cannot be given on initialization statements (CARRY, MDCODES, NAME).

- To continue a statement onto another line, enter a dash at the end of the line and continue from any position on the next line.

- The maximum line length is 255 characters and the maximum total number of characters for a statement is 1024 excluding continuation dashes and trailing blanks after the dash.

4.2 Sample Set of Recode Statements

To give some idea of how the elements of the Recode language fit together, a sample set of Recode statements is given below.

```
$RECODE
   IF V5 LT 8 THEN REJECT                    (exclude cases where V5 < 8)
   IF NOT MDATA(V6) THEN R51=TRUNC(V6/4) -
      ELSE R51=0
   R52=BRAC(V10,0-24=1,25-49=2,50-74=3, -    (group values of V10)
      74-99=4,TAB=1)
   R53=BRAC(V11,TAB=1)                       (group V11 the same way as V10)
   IF V26 INLIST(1-10) THEN R54=1 AND -
      R55=1 ELSE R54=2
   IF R54 EQ 1 THEN GO TO L1
   R55=99
   R56=V15 + V35
   GO TO L2
L1 R56=99
L2 R57=COUNT(1,V20-V27,V29)                  (count how many of the listed
                                              variables have the value 1)
   NAME R52 'GROUPED AGE',  -
        R53 'GROUPED AGE AT MARRIAGE'
   MDCODES R55(99),R56 (99)
```

4.3 Missing Data Handling

Except in the special functions MAX, MEAN, MIN, STD, SUM, VAR, Recode does not automatically check the values of variables for missing data. The user must therefore control specifically for missing data before doing calculations with variables. The MDATA function is available for this purpose; e.g.

```
IF MDATA (V5,V6) THEN R1=999 ELSE R1=V5+V6
```

There are two additional functions, MD1 and MD2, which return the 1st or 2nd missing data code value for a variable; e.g.

```
R2=MD1(V6)
```

assigns R2 the value of the 1st missing data code of V6.

Finally, missing data codes can be assigned to R or V variables with the MDCODES definition statement; e.g.

```
MDCODES R3(8,9)
```

assigns 8 and 9 as the 1st and 2nd missing data codes for R3.

Sometimes a set of Recode statements does not assign a value to an R-variable for a particular data record. The R-variable will then take the default MD1 value of 1.5×10^9 to which it is initialized. To change this to a more acceptable missing data value, we must test if the value is large and, if so, assign an appropriate missing data value, e.g.

```
IF R100 GT 1000000 THEN R100=99
MDCODES R100(99)
```

4.4 How Recode Functions

Syntax checking and interpretation. Recode statements are read and analyzed for errors prior to interpretation of other IDAMS program control statements and prior to program execution. If errors are found, diagnostic messages are printed and execution of the program is terminated.

Results. Recode prints out the Recode statements input by the user along with syntax errors detected if any. This occurs before the program is executed, i.e. before the interpretation of the program control statements is printed.

Initialization before starting to process the Data file. If there are no syntax errors, tables, missing data codes, names, etc. are initialized (according to the initialization/definition statements supplied by the user) before starting to read the data. R-variables in CARRY statements are initialized to zero.

Initialization before processing each data case. At the start of processing of each case and before execution of the Recode statements for that case, all R-variables, except those listed in CARRY statements, are initialized to the IDAMS internal default missing data value (1.5×10^9).

Execution of Recode statements. The actual recoding takes place after the data for a case is read and after the main filter has been applied. Cases not passing the filter are not passed to the recoding routines. Recode variables cannot therefore be used in main filters.

The use of the Recode statements is sequential (i.e. the first statement is used first, then the second, third, etc.) except as modified by GO TO, BRANCH, RETURN, REJECT, ENDFILE, ERROR statements (the control statements). When all statements have been used, the case is passed to the IDAMS program being executed.

When the IDAMS program has finished using the case, the next case passing the main filter is processed, the R-variables (except the CARRY variables) being reinitialized to missing data and the Recode statements executed for that case and so on until the end of the data file is reached.

Testing Recode statements. Errors in logic can be made which are not detectable by the Recode facility. To check the intended results against those generated by Recode, the Recode statements should be tested on a few records using the LIST program with the parameter MAXCASES set, say, to 10. The data values

for the variables input and the corresponding result variables can then be inspected.

Files used by Recode. When a $RECODE command is encountered in the Setup file, subsequent lines are copied into a work file on unit FT46. The RECODE program reads Recode statements from this file and analyzes them for errors prior to interpretation of other IDAMS program control statements and prior to program execution. If errors are found, diagnostic messages are printed and execution of the entire IDAMS step is terminated.

Interpreted statements are written in the form of tables to a work file on unit FT49 from where they are read by the IDAMS program being executed.

Messages about Recode statements are written to unit FT06 along with results from the IDAMS program being executed.

4.5 Basic Operands

Variables. Variables in Recode refer either to input variables (V-variables) or result variables (R-variables). They are defined as follows:

> Input variables (Vn). "V" followed by a number. These are variables as defined by the input dictionary. Their values may be changed by Recode (e.g. V10=V10+V11). Variables should normally be numeric but alphabetic variables of not more than 4 characters can also be used, in particular, they can be recoded to numeric values.
>
> Result variables (Rn). "R" followed by a number (1 to 9999). These are variables that are created by the user. R-variables (except for those listed in CARRY statements - see below) are initialized to the default missing value of 1.5×10^9 before processing of each case.
>
> To use an R-variable in a program, specify an R (instead of V) on the variable list attached to a keyword parameter (e.g. WEIGHT=R50 or VARS=(R10-R20)). When printed out by programs, a result variable number is sometimes identified by a negative sign. Thus, variable "10" is V10 and variable "-10" is R10. It is less confusing to use numbers for the result variables which are distinct from input variable numbers. R-variables are always numeric.

Numeric constants. Constants may be integer or decimal, positive or negative, e.g. (3, 5.5, -50, -0.5).

Character constants. Character constants are enclosed in single primes (e.g. 'ABCXYZ', 'M'). A prime within a character constant must be represented by two adjacent primes (e.g. DON'T would be written: 'DON"T'). Character constants are used in the NAME statement to assign names to new variables. They can also be used in logical expressions to test values of alphabetic variables (e.g. IF V10 EQ 'M'); only the first 4 characters are used in such comparisons and constants/variables values of length < 4 are padded on the right with blanks. Character constants cannot be used in arithmetic functions (except BRAC).

4.6 Basic Operators

Arithmetic operators. Arithmetic operators are used between arithmetic operands. Available operators, in precedence order, are:

-	(negation)
EXP x	(exponentiation to the power x, where $-181 < x < 175$)
*	(multiplication)
/	(division)
+	(addition)
-	(subtraction)

Relational operators. Relational operators are used to determine whether or not two arithmetic values have a particular relationship to one another. The relational operators are:

LT	(less than)
LE	(less than or equal)
GT	(greater than)
GE	(greater than or equal)
EQ	(equal)
NE	(not equal)

Logical operators. Logical operators are used between logical operands. Logical operands take only the values "true" or "false". These are:

NOT	
AND	(both)
OR	(either)

4.7 Expressions

An expression is a representation of a value. A single constant, variable, or function reference is an expression. Combinations of constants, variables, functions and other expressions with operators are also expressions. Recode can evaluate arithmetic and logical expressions. Note that brackets can be used anywhere in an expression to clarify the order in which it is to be evaluated.

Arithmetic expressions. Arithmetic expressions are created using arithmetic operators and variables, constants and arithmetic functions. They yield a numeric value. Examples are:

```
V732              (the value of V732)
44                (the constant 44)
R67/V807 + 25     (25 plus the value of R67 divided by the value of V807)
LOG(R10)          (the log of the value of R10)
```

Logical expressions. Logical expressions are evaluated to a "true" or "false" value. Logical variables do not exist in the Recode language, so that the result of logical expressions cannot be assigned to a variable. Logical expressions can only be used in IF statements. Examples are:

```
R5 EQ V333
```

True if the value of R5 is equal to the value of V333, and false otherwise.

```
(V62 GT 10) OR (R5 EQ V333)
```

True if either of the logical expressions results in a true value, and false if both result in a false value.

```
MDATA(V10,R20) AND V9 GT 2
```

True if the value of V10 or the value of R20 is a missing data code and the value of V9 is larger than 2, false otherwise.

4.8 Arithmetic Functions

Arithmetic functions all return a single numeric value. The argument list for functions can be simple lists enclosed in parentheses or highly structured lists involving both keyword elements and elements in specific positions in the list. The available functions are:

Function	*Example*	*Purpose*
ABS	`ABS(R3)`	Absolute value
BRAC	`BRAC(V5,TAB=1,ELSE=9, -` `1-10=1,11-20=2)`	Univariate grouping
	`BRAC(V10,'F'=1,'M'=2)`	Alphabetic recoding
COMBINE	`COMBINE V1(2), V42(3)`	Combination of 2 variables
COUNT	`COUNT(1,V20-V25)`	Counting occurrences of a value across a set of variables
LOG	`LOG(V2)`	Logarithm to the base 10
MAX	`MAX(V10-V20)`	Maximum value
MD1,MD2	`MD1(V3)`	Value of missing data code
MEAN	`MEAN(V5-V8,MIN=2)`	Mean value
MIN	`MIN(V10-V20)`	Minimum value
NMISS	`NMISS(V3-V6)`	Number of missing data values
NVALID	`NVALID(V3-V6)`	Number of non-missing values
RAND	`RAND(0)`	Random number
RECODE	`RECODE V7,V8,(1/1)(1/2)=1, -` `(2-3/3)=2, ELSE=0`	Multivariate recoding
SELECT	`SELECT (BY=V10,FROM=R1-R5,9)`	Selecting the value of one of a set of variables according to an index variable
SQRT	`SQRT(V2)`	Square root
STD	`STD(V20-V25,MIN=4)`	Standard deviation
SUM	`SUM(V6,V8,V9-V12,MIN=3)`	Sum of values
TABLE	`TABLE(V5,V3,TAB=2,ELSE=9)`	Bivariate recoding
TRUNC	`TRUNC(V26/3)`	Integer part of the argument's value
VAR	`VAR(V6,R5-R10,MIN=7)`	Variance

The exact syntax for each function is given below.

ABS. The ABS function returns a value which is the absolute value of the argument passed to the function.

Prototype: ABS(arg)

Where arg is any arithmetic expression for which the absolute value is to be taken.

Example:

```
R5=ABS(V5-V6)
```

BRAC. The BRAC function returns a value which is derived from performing specified operations (rules) upon a single variable.

Prototype: BRAC(var [,TAB=i] [,ELSE=value] [,rule1,...,rule n])

Where:

- var is any V- or R-type variable whose values are being tested.
- TAB=i either numbers the set of rules and the associated ELSE established in this use of BRAC (optional), or references a set of rules established in a previous use of BRAC. Note: The ELSE clause is considered part of the set of rules.
- ELSE=value is used when the value of var cannot be found in the rules given. If ELSE=value is omitted, ELSE=99 is assumed, i.e. BRAC always recodes.
- rule1, rule2,...,rule n are the set of rules defining the values to be returned depending on the value of var. The rules are expressed in the form: x=c, where x defines one or more codes and c is the value to be returned when the value of var equals the code(s) defined by x. The possible rules (where m is any numeric or character constant) are:

  >m=c (if the value of var is greater than m, return value c).

  <m=c (if the value of var is less than m, return value c).

m=c (if the value of var is equal to m, return value c).

m1-m2=c (if the value of var is in the range m1 to m2, i.e. m1<=var<=m2, return value c).

- As many rules may be given as necessary. They are evaluated from left to right, and the first one which is satisfied is used. Note that ">" and "<" are used, not the GT and LT logical operators.
- ELSE, TAB, and the rules may be specified in any order.
- Ranges of alphabetic values, e.g. 'A'-'C', are not allowed.

Examples:

```
R1=BRAC(V10,TAB=1,ELSE=9,1-10=1,11-20=2,<0=0)
```

The value of R1 will be 1 if variable 10 is in the range 1 to 10, 2 if V10 is in the range 11 - 20, and 0 if V10 is less than 0. If V10 has any other value, e.g. -3, 10.5, 25, 0, then the ELSE clause would be applied, and R1 would be 9. These bracketing rules are labelled table 1 so they can be re-used, e.g.

```
R2=V1 + BRAC(V2, TAB=1) * 3
```

In this example V2 would be bracketed by the same rules as for V10 in the previous example. R2 would be set to V1 + (the result of bracketing multiplied by 3).

```
R100=BRAC(V10,'F'=1,'M'=2,ELSE=9)
```

This is an example of recoding an alphabetic variable, which has values 'F' or 'M', to numeric values of 1 and 2.

COMBINE. The COMBINE function returns a unique value for each combination of values of the variables that are used as arguments. This function is normally used with categorical variables.

Prototype: COMBINE var1(n1), var2(n2),...,varm(nm)

Where:

- var1 to varm are the V- or R-variables to combine.
- n1 to nm are the maximum codes +1 of the respective variables.
- The list of arguments to the COMBINE function is not enclosed in parentheses.
- Each variable must have only non-negative and integer values.
- The values returned are computed by the following formula:

  V1 + (n1 * V2) + (n1 * n2 * V3) + (n1 * n2 * n3 * V4) etc.

  The user, however, would normally determine the result of the function by listing the combinations of values in a table as in the first example below.

Examples:

```
R1=COMBINE V6(2), R330(3)
```

Assume that V6 has two codes (0,1) representing men and women respectively and R330 has three codes (0,1,2) representing young, middle aged and old respondents, the statement will combine the codes of V6 and R330 to give a single variable R1 as follows:

V6	V330	R1	
0	0	0	Young men
1	0	1	Young women
0	1	2	Middle aged men
1	1	3	Middle aged women
0	2	4	Old men
1	2	5	Old women

Since V6 has two codes, and R330 has three, R1 will have six. In the above example, if V6 had codes 1 and 2 instead of 0 and 1, the maximum value should be stated as "3". This would allow for the values of 0,1, and 2, although code 0 would never appear. To avoid these "extra" codes, the user should first recode such variables to give a contiguous set of codes starting from 0, e.g. BRAC(V6,1=0,2=1).

Restrictions:

- There may be up to 13 variables.
- The COMBINE function cannot be used with other functions in the same assignment statement.
- Care should be taken to accurately specify the maximum codes when using the COMBINE function. Otherwise, non-unique values will be generated. For example, with "COMBINE V1(2), V2(4)" the function will return a value of 7 for the pair of values, V1=1 and V2=3, and will also return a value of 7 for the pair of values V1=3 and V2=2. If values of 3 might exist for V1, then n1 should be specified as 4 (1 + maximum code).

COUNT. The COUNT function returns a value which is equal to the number of times the value of a variable or constant occurs as the value of one of the variables in the list "varlist".

Prototype: COUNT(val,varlist)

Where:

- val is normally a constant but can also be a V- or R-variable.
- varlist gives the V- and/or R-variables whose values are to be checked against val.

Examples:

```
R3=COUNT(1,V20-V25)
```

R3 will be assigned a value equal to the number of times the value 1 occurs in the 6 variables V20-V25. This might be used for example to count the number of "YES" responses by a respondent to a set of questions.

```
R5=COUNT(V1,V8-V10)
```

R5 will be assigned a value equal to the number of times that the value of V1 occurs also as the value of variables V8-V10.

LOG. The LOG function returns a floating-point value which is the logarithm to the base 10 of the argument passed to the function.

Prototype: LOG(arg)

Where arg is any arithmetic expression for which the log to the base 10 is to be taken.

Examples:

```
R10=LOG(V30)
```

Note: The logarithm of any number X to any other base B can readily be found by the following simple transformation:

```
R1=LOG(X)/LOG(B)
```

For the natural logarithm (base e), this becomes simply: R1=2.302585 * LOG(X).

Thus R1=2.302585 * LOG(V30) will assign to R1 the natural logarithm of variable 30.

MAX. The MAX function returns the maximum value in a set of variables. Missing data values are excluded. The MIN argument can be used to specify the minimum number of valid values for a maximum to be calculated. Otherwise the default missing data value 1.5×10^9 is returned.

Prototype: MAX(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the maximum value. n defaults to 1.

Example:

```
R12=MAX(V20-V25)
```

MD1, MD2. The MD1 (or MD2) function returns a value which is the first (or second) missing data code of the variable given as the argument.

Prototype: MD1(var) or MD2(var)

Where var is any input variable (V-variable) or previously defined result variable (R-variable).

Example:

```
R12=MD2(V20)
```

For each case processed, R12 will be assigned the second missing data code for input variable V20.

MEAN. The MEAN function returns the mean value of a set of variables. Missing data values are excluded. The MIN argument can be used to specify the minimum number of valid values for a mean to be calculated. Otherwise the default missing value 1.5×10^9 is returned.

Prototype: MEAN(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the mean value. n defaults to 1.

Example:

```
R15=MEAN(R2-R4,V22,V5,MIN=2)
```

The result will be the mean of the specified variables, if at least two of the variables have non-missing values. Otherwise, the result will be 1.5×10^9.

MIN. The MIN function returns the minimum value in a set of variables. Missing data values are excluded. The MIN argument can be used to specify the minimum number of valid values for a minimum to be calculated. Otherwise the default missing value 1.5×10^9 is returned.

Prototype: MIN(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the minimum value. n defaults to 1.

Example:

```
R10=MIN(V5,V7,V9,R2)
```

NMISS. The NMISS function returns the number of missing values in a set of variables.

Prototype: NMISS(varlist)

Where varlist is a list of V- and R-type variables.

Example:

```
R22=NMISS(R6-R10)
```

The returned value depends on how many of the variables R6 - R10 have missing values. The maximum value is 5 for a case in which all 5 variables have missing data.

NVALID. The NVALID function returns the number of valid values (non-missing values) in a set of variables.

Prototype: NVALID(varlist)

Where varlist is a list of V- and R-type variables.

Example:

```
R2=NVALID(V20,V22,V24)
```

The returned value depends on how many of the variables have valid values. The maximum value of 3 will be obtained if all 3 variables have valid values. 0 will be returned if all 3 are missing.

RAND. The RAND function returns a value which is a uniformly distributed random number based upon the arguments "starter" and "limit" as described below.

Prototype: RAND(starter [,limit])

Where:

- starter is an integer constant that is used to initiate the random sequence. If starter is 0, then the current clock time is used.
- limit is an optional argument. It is an integer constant that is used to specify the range (i.e. 3 means a range of 1 to 3). The default value is 10, which means the default range is 1 to 10.

Examples:

```
R1=RAND(0)
IF RAND(0) NE 1 THEN REJECT
```

For each case processed, R1 will be set equal to a random number, uniformly distributed from 1 to 10. The sequence is initialized to the clock time the first time RAND is executed. Note that RAND can be used with the REJECT statement to select a random sample of cases. The 2nd example will result in including a random 1/10 sample of cases.

RECODE. The RECODE function is used to return one value based upon the concurrent values of m variables.

Prototype: RECODE var1,var2,...,varm [,TAB=i] [,ELSE=value] [,rule1,rule2,...,rule n]

Where:

- var1,var2,...,varm is a list of up to 12 V and/or R variables to be tested.
- TAB=i either numbers the set of recode rules established in this use of RECODE (optional) or references a set of rules established in a previous use of RECODE. Note: the ELSE value **is not** considered a part of the set of recode rules.
- ELSE=value (optional) indicates the value to be returned if none of the code lists match the values of the variables. While it is usually a constant, the value may be any arithmetic expression. If ELSE is omitted, and none of the code lists match the variable values, the function does not return a value, i.e. the value of the result variable is left unchanged. If this is the first assignment statement for a variable, then its value will be the input data value for a V-variable or missing data for an R-variable.
- rule1, rule2,..., rule n are the set of rules defining the values to be returned depending on the values of var1, var2,..., varm. Each rule is of the form "(code list 1) (code list 2) ... (code list p)=c". Each code list is of the form "(a1/a2/.../am)" where a1 is the code to be compared with var1, a2 is the code to be compared with var2, etc. Here c is the value to be returned when var1,var2,..., varm match the codes defined in any of the code lists.

The prototype for a rule is:

```
(a1/a2/.../am)(b1/b2/.../bm)...(x1/x2/.../xm)=c
```

Each code list contains a list and/or a range of values for every variable, e.g. with two variables, (3/2)(6-9/4)(0/1,3,5)=1.
The codes in the code list may be separated by a slash (indicating "AND") or by a vertical bar (indicating "OR"), although only one or the other may be used in any given code list.
For example:

```
(a1/a2/a3)=c
(the function will return c if var1=a1 and var2=a2 and var3=a3)

(a1|a2|a3)=c
(the function will return c if var1=a1 or var2=a2 or var3=a3)
```

- Rules are examined from left to right. The first code list which matches the variable list values determines the value to be returned.
- The argument list for the RECODE function is not enclosed in parentheses.
- TAB, ELSE and rules may be in any order.

Examples:

```
R7=RECODE V1,V2,(3/5)(7/8)=1,(6-9/1-6)=2
```

R7 will be assigned a value based on the values of V1 and V2. In this example, R7 will be set to 1 if V1=3 and V2=5, or if V1=7 and V2=8. R7 will be set to 2 if V1=6-9 and V2=1-6. In all other instances, R7 will be unchanged (see above).

```
R7=RECODE V1,V2,TAB=1,ELSE=MD1(R7),(3/5)(7/8)=1,(6-9/1-6)=2
```

R7 will be assigned a value the same as in the preceding example, except that R7 will be set equal to its MD1 value when the rules are not met. The TAB=1 will allow these rules to be used in another RECODE function call.

Restriction: When the RECODE function is used, it must be the only operand on the right-hand side of the equals sign.

SELECT. The SELECT function returns the value of the variable or constant in the FROM list holding the same position as the value of the BY variable. (Warning: If the value of the BY variable is less than 1 or greater than the number of variables in the FROM list, a fatal error results). There may be up to 50 items in the FROM list. The maximum value of the BY variable is therefore 50. A SELECT function may be combined with other functions, operations, and variables to form a complex expression. Note: The SELECT **function** selects the **value** of one of a set of variables; the SELECT **statement** selects the **variable** to be used for the result. (See section "Special Assignment Statements" for description of SELECT statement).

Prototype: SELECT (FROM=list of variables and/or constants, BY=variable)

Example:

```
R10=SELECT (FROM=R1-R3,9,BY=V2)
```

R10 will take the value of R1, R2, R3 or 9 for values of 1, 2, 3 or 4 respectively of V2.

SQRT. The SQRT function returns a value which is the square root of the argument passed to the function.

Prototype: SQRT(arg)

Where arg is any arithmetic expression.

Example:

```
R5=SQRT(V5)
```

STD. The STD function returns the standard deviation of the values of a set of variables. Missing data values are excluded. The MIN argument can be used to specify the minimum number of valid values for a standard deviation to be calculated. Otherwise the default missing value 1.5×10^9 is returned.

Prototype: STD(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the standard deviation. n defaults to 1.

Example:

```
R5=STD(V20-V24,R56-R58,MIN=3)
```

SUM. The SUM function returns the sum of the values of a set of variables. Missing values are excluded. The MIN argument can be used to specify the minimum number of valid values for a sum to be calculated. Otherwise the default missing value 1.5×10^9 is returned.

Prototype: SUM(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the sum. n defaults to 1.

Example:

```
R8=SUM(V20,V22,V24,V26,MIN=3)
```

If three or more of the variables have valid values, the sum of these is returned. Otherwise the value 1.5×10^9 is returned.

TABLE. The TABLE function returns a value based on the concurrent values of two variables.

Prototype: TABLE (r, c, [TAB=i,] [ELSE=value,] [PAD=value,] COLS c1,c2,...,cm,

ROWS r1(row r1 values),r2(row r2 values),...,rn(row rn values))

Where:

- r is a variable or constant that will be used as a "row index" to a table.
- c is a variable or constant that will be used as a "column index" to a table.
- TAB=i either numbers the table defined in this use of TABLE (optional) or references a table defined in a previous use of TABLE.
- ELSE=value gives a value to use for pairs of values that are not defined in the table. The value may be an arithmetic expression. The value of ELSE defaults to 99 if not specified, i.e.TABLE always returns a value.
- PAD=value gives a value to be inserted into any cell which is defined by the COLS specifications but not defined by the ROWS specifications.
- TAB, ELSE and PAD may be specified in any order.
- c1,c2,...,cm are the columns of the table. Ranges may be used in the column definitions.
- r1,r2,...,rn are the rows of the table. The total size of the table will be m by n, where m is the number of columns and n is the number of rows.
- (row r1 values), (row r2 values),...,(row rn values) are the values returned depending on the values of r and c. The values are given in the same order as the column specifications; the first value corresponds to c1, the second to c2, etc. Ranges may be used in the row value definitions.

Examples: Assume the following table:

	Col:	1	2	3	4	5	6
Row:	2	1	1	2	2	3	4
	3	1	2	2	2	3	4
	5	1	2	2	2	3	4
	6	3	3	3	3	3	4
	8	9	9	9	9	9	9

```
R1=TABLE (V6, V4, TAB=1, ELSE=0, PAD=9, COLS 1-6, ROWS 2(1,1,2,2,3,4), -
   3(1,2,2,2,3,4),5(1,2,2,2,3,4),6(3,3,3,3,3,4),8(9))
```

If V6 equals 5 and V4 equals 3, then R1 will be assigned the value 2 (intersect of row 5 and column 3).
If V6 equals 2 and V4 equals 6, then R1 will be assigned the value 4 (intersect of row 2 and column 6).
If V6 equals 4 and V4 equals 2, then R1 will be assigned the value 0 (row 4 is not defined; the ELSE value is used).

```
R5=TABLE (3, V8, TAB=7, ELSE=TABLE(V1,V8,TAB=1) )
```

This will use the table named "7" with 3 as the row index and the value of V8 as the column index. If a value of V8 is not in table 7 then the table "1" will be used with row index V1 and column index V8.

TRUNC. The TRUNC function returns the integer value of an argument.

Prototype: TRUNC(arg)

Where arg is any arithmetic expression for which the integer value is to be taken.

Example:

```
R5=TRUNC(V5)
```

R5 will be assigned the value of the input variable V5 truncated to an integer.

VAR. The VAR function returns the variance of the values of a set of variables, excluding missing data. The MIN argument can be used to specify the minimum number of valid values for the variance to be calculated. Otherwise the default missing value 1.5×10^9 is returned.

Prototype: VAR(varlist [,MIN=n])

Where:

- varlist is a list of V- and R-type variables, and constants.
- n is the minimum number of valid values for computation of the variance. n defaults to 1.

Example:

```
R9=VAR(V5-V10)
```

4.9 Logical Functions

Logical functions return a value of "true" or "false" when evaluated. They cannot be used as arithmetic operands. Logical functions are used in logical expressions and logical expressions comprise the test portion of conditional "IF test THEN..." statements. The available functions are:

Function	*Example*	*Purpose*
EOF	`IF EOF THEN GO TO NEXT`	Checks for the end of the data file
INLIST	`IF V5 INLIST(2,4,6) THEN - R100=1 ELSE R100=0`	Searches a list of values
MDATA	`IF MDATA(V5,V6) THEN R101=99`	Checks for missing data

EOF. The EOF function is used for aggregation of values across cases. See example 10 in section "Examples of Use of Recode Statements". The presence of the EOF function causes the Recode statements to be executed once more after the end-of-file has been encountered. The value of the EOF function is true during this after-end-file pass of the Recode statements and is false at all other times.

For the final pass through the Recode statements, V-variables will have the value they had after the last case was fully processed. R-variables (except those listed in CARRY statements) will be reinitialized to 1.5×10^9. CARRY R-variables will be left untouched. The user must be careful to set up a correct path to be followed through the Recode statements when end-of-file is reached.

Prototype: EOF

Example:

```
IF R1 NE V1 OR EOF THEN GO TO L1
```

INLIST. The INLIST function (abbreviated IN) returns a value of "true" if the result of an arithmetic expression is one of a specified set of values. If the expression equals a value outside the set of values, the function returns a value of "false".

Prototype: expr INLIST(values) or expr IN(values)

Where:

- expr is any arithmetic expression or a single variable.
- values is a list of values. These may be discrete and/or value ranges.

Examples:

```
IF R12 INLIST(1-5,9,10) THEN V5=0
```

If R12 has a value of 1,2,3,4,5,9 or 10, the INLIST function returns a value of "true", and input variable V5 is set to 0. Otherwise, INLIST returns a value of "false" and input variable V5 retains its original value.

```
IF (V3 + V7) IN(2,4,5,6) THEN R1=1 ELSE R1=9
```

If the sum of input variables V3 and V7 results in the value 2,4,5, or 6, then INLIST returns a value of "true" and result variable R1 will contain the value 1. Otherwise, INLIST returns a value of "false" and R1 will be set to 9.

MDATA. The MDATA function returns a value of "true" if any of the variables passed to the function have missing data values; otherwise, the function returns a value of "false". This function is used quite often, since missing data is not automatically checked in the evaluation of expressions except in the MAX, MEAN, MIN, STD, SUM and VAR functions.

Prototype: MDATA(varlist)

Where varlist is a list of V- and R-variables. There can be a maximum of 50 variables in this list.

Example:

```
IF MDATA(V1,V5-V6) THEN R1=MD1(R1) ELSE R1=V1+V5+V6
```

If any variable in the list V1, V5, V6 has a value equal to its MD1 code or in the range specified by its MD2 code, the MDATA function will return a value of "true", and result variable R1 will be set to its first missing data code. Otherwise, the MDATA function will return a value of "false" and R1 is set to the sum of V1, V5, V6.

4.10 Assignment Statements

These are the main structural units of the Recode language. They are used to assign a value to a result. Any number between 1 and 9999 may be used for an R-variable but it avoids confusion if the R-numbers are distinct from V-numbers of variables in the input dictionary, e.g. if there are 22 variables in the dictionary then start numbering R-variables from R30. Assignment statements can also be used to assign a new value

to an input variable. In this case the original value of the input variable is lost for the duration of the particular IDAMS program execution.

Prototype: variable=expression

Where:

- variable is any input (Vn) or result (Rn) variable.
- expression is any arithmetic expression optionally using Recode arithmetic functions.
- Note that variables used in the expression are not automatically checked for missing data except in the special functions MAX, MEAN, MIN, STD, SUM, VAR. In all other cases, specific statements to check for missing data must be introduced where appropriate. See below under "Conditional statements" for example.

Examples:

```
R10=5
```

R10 is assigned the constant 5 as its value.

```
R5=2*V10 + (V11 + V12)/2
```

Any arithmetic expression may be used and parentheses are used to change normal precedence of the arithmetic operators.

```
V20=SQRT(V20)
```

The value in V20 is replaced by its square root using the SQRT function.

```
R20=BRAC(V6,0-15=1,16-25=2,26-35=3,36-90=4,ELSE=9)
```

R20 is assigned the value 1, 2, 3, 4 or 9 according to the group into which the value of V6 falls.

```
R10=MD1(V10)
```

R10 is assigned a value equal to V10's first missing data code.

4.11 Special Assignment Statements

DUMMY. The DUMMY statement produces a series of "dummy variables", coded 0 or 1, from a single variable.

Prototype: DUMMY var1,...,varn USING var(val1)(val2)...(valn)[ELSE expression]

Where:

- var1, var2,...,varn is a list of the dummy variables whose values are defined by this statement. They may be V- or R-variables, may be listed singly or in ranges, and must be separated by commas (e.g. R1-R3, R10, R7-R9, V20). The order specified is preserved.
- Double references (R1, R3, R1) are valid.
- var is any V- or R-variable. The value of this variable is tested against the value lists (val1)(val2) etc. to set the appropriate value of the dummy variables.
- (val1)(val2)...(valn) are lists of values used to set the values of the dummy variables. There must be the same number of lists as dummy variables (var1, var2, ..., varn). Value lists can contain single constants or ranges or both.
- expression is any arithmetic expression that is used as the value for all dummy variables when the value of the variable var is not in one of the lists of values. Expression defaults to the constant 0.

- The value of the variable var is tested against the value lists (the number of value lists must equal the number of dummy variables); if var has a value in the first value list, the first dummy variable is set to 1, the others to 0; if the var value occurs in the second value list, the second dummy variable is set to 1, the others to 0, etc. If the var value occurs in none of the value lists, all dummy variables are set to the value specified after the ELSE (defaults to 0).

Example:

```
DUMMY R1-R3 USING V8(1-4)(5,7,9)(0,8) ELSE 99
```

The following chart shows the values of R1, R2 and R3 based on different V8 values:

```
V8:   1   2   3   4   5   7   8   9   0   OTHER
R1:   1   1   1   1   0   0   0   0   0   99
R2:   0   0   0   0   1   1   0   1   0   99
R3:   0   0   0   0   0   0   1   0   1   99
```

SELECT. The SELECT statement causes the **variable** in the FROM list holding the same position as the value of the BY variable to be set equal to the value of the expression to the right of the equals sign i.e. it selects **which** variable is to be assigned a value. If the value of the BY variable is less than 1 or greater than the number of variables in the FROM list, a fatal error results. The maximum number of items in the FROM list is 50. Therefore the maximum value of the BY variable is 50.

Prototype: SELECT (FROM=variable list, BY=variable)=expression

Examples:

```
SELECT (FROM=R1,V3-V10, BY=R99)=1
SELECT (BY=V1, FROM=V8,R2,R5)=R7*5
```

In the first example, R1 will be set to 1 if R99 equals 1; V3 will be set to 1 if R99 equals 2; ... ; and V10 will be set to 1 if R99 equals 9. If R99 is greater than 9 or less than 1, a fatal error will result. The values of the eight variables not selected will not be altered.

SELECT may be used to form a loop as follows:

```
    R99=1
L1  SELECT (BY=R99, FROM=R1,V3-V10)=0
    IF R99 LT 9 THEN R99=R99+1 AND GO TO L1
```

The nine variables R1, V3-V10 will be set to zero, one after another, as R99 is incremented from 1 to 9. The loop is completed when R99 equals 9 and all variables have been initialized.

4.12 Control Statements

Recode statements are normally executed on each data case in order from first to last. The order can be changed with one of the control statements:

Statement	*Example*	*Purpose*
BRANCH	`BRANCH (V16,L1,L2)`	Branch depending on the value of a variable
CONTINUE	`CONTINUE`	Continue with next statement
ENDFILE	`ENDFILE`	Do not process any more data cases after this one
ERROR	`ERROR`	Terminate execution completely
GO TO	`GO TO TOWN`	Branch unconditionally
REJECT	`REJECT`	Reject the current data case
RELEASE	`RELEASE`	Release the current data case to the program for processing and then execute recode statements again without reading another case
RETURN	`RETURN`	Use the current case for analysis with no further recoding

BRANCH. The BRANCH statement changes the sequence in which statements are executed, depending on the value of a variable.

Prototype: BRANCH(var,labels)

Where:

- var is a V or R-variable.
- labels is a list of one or more 1 to 4-character statement labels.

Example:

```
BRANCH(R99,LAB1,LAB2,LAB3)
```

Transfer is made to LAB1, LAB2, or LAB3, depending on whether R99 has a value of 1,2, or 3.

CONTINUE. CONTINUE is a simple statement which performs no operation. It is used as a convenient transfer point.

Prototype: CONTINUE

Example:

```
        IF V17 EQ 10 THEN GO TO AT
        R10=V11
        GO TO THAT
AT      R20=V11*100
THAT    CONTINUE
```

ENDFILE. The ENDFILE statement causes the Recode facility to close the input dataset exactly as if an end-of-file had been reached. If the EOF function has been specified, the EOF function will be given a true value for a final pass through the Recode statements from the beginning, after ENDFILE has been executed.

Prototype: ENDFILE

Example:

```
IF V1 EQ 100 THEN ENDFILE
```

This statement can be used to test a set of Recode statements or an IDAMS setup on the first n cases of a dataset.

ERROR. The ERROR statement directs the Recode facility to terminate execution with an error message that indicates the number of the case and the number of the Recode statement at which the error occurred.

Prototype: ERROR

Example:

```
      IF R6 EQ 2 THEN GO TO B
      ERROR
B     CONTINUE
```

GO TO. The GO TO statement is used to change the sequence in which the statements are executed. In the absence of a GO TO or a BRANCH statement, each statement is executed sequentially.

Prototype: GO TO label

Where label is a 1-4 character statement label. The statement identified by the label may be physically before or after the GO TO statement. (Warning: Be careful of referencing a statement before the GO TO, as an endless loop can be formed).

Example:

```
        GO TO TOWN
        .
        .
        R10=R5
        GO TO 1
TOWN    R10=R5+V11
1       R11=...
```

REJECT. The REJECT statement directs the Recode facility to reject the present case and obtain another case. The new case is then processed from the beginning of the Recode statements. Thus, REJECT can be used as a filter with R-variables.

Prototype: REJECT

Example:

```
IF MDATA (V8,V12-V13) THEN REJECT
```

RELEASE. The RELEASE statement directs the Recode facility to release the present case to the program for processing and to regain control after the processing without reading another case. After regaining control, Recode resumes with the first Recode statement. RELEASE can be used to break up a single record into several cases for analysis. Note: When using the RELEASE statement, care should be taken that processing will not continue indefinitely.

Prototype: RELEASE

Example:

```
CARRY (R1)
R1=R1+1
IF R1 LT V1 THEN RELEASE ELSE R1=0
```

RETURN. The RETURN statement directs the Recode facility to return control to the IDAMS program. No other Recode statements are executed for the current case.

Prototype: RETURN

Example:

```
      IF V8 LT 12 THEN GO TO A
      RETURN
A     R10=V8
```

4.13 Conditional Statements

The **IF** statement allows conditional assignment and/or conditional control. It is a compound statement with several simple statements connected by the keywords THEN, AND and ELSE.

Prototype:

IF test THEN stmt1 [AND stmt2 AND ... stmt n][ELSE estmt1] [AND estmt2 AND ... estmt n]

Where:

- test may be any combination of logical expressions (including logical functions) connected by AND or OR and optionally preceded by NOT. It may be, but need not be, enclosed in parentheses.
- stmt1,...,stmt n,estmt1,...,estmt n may be any assignment or control statement (except CONTINUE).
- The statement(s) between the THEN and ELSE are executed if the test is true.
- The statement(s) after the ELSE are executed if the test is false. If no ELSE clause is present, the next statement is executed.

- The THEN and ELSE keywords may each be followed by any number of statements, each connected by the keyword AND.

Examples:

```
IF V5 EQ V6 THEN R1=1 ELSE R1=2
```

Set R1 to 1 if the value of V5 equals the value of V6; otherwise set R1 to 2.

```
IF MDATA(V7,V10-V12) THEN R6=MD1(V7) AND R10=99 -
   ELSE R6=V7+V10+V11 AND R10=V12*V7
```

Set R6 to V7's first missing data value and R10 to 99 if any of the variables V7, V10, V11, V12 are equal to their missing data codes. Otherwise set R6 equal to the sum of V7, V10 and V11, and also set R10 equal to the product of V12 and V7.

```
IF (V5 NE 7 AND R8 EQ 9) THEN V3=1 ELSE V3=0
```

Set V3 to 1 if both V5 is not equal to 7 and R8 is equal to 9. (Note: The parentheses are not required).

```
IF MDATA(V6) OR V10 LT 0 THEN GO TO X
```

If the value of V6 is missing or V10 is less than 0, branch to the statement labelled X; otherwise continue with the next statement.

4.14 Initialization/Definition Statements

These statements are executed once, before processing of the data starts, to initialize values to be used during the execution of Recode statements. They cannot be used in expressions and they cannot have labels.

CARRY. The CARRY statement causes the values of the variables listed to be carried over from case to case. CARRY variables are initialized only once (before starting to read the data) to zero. The CARRY variables can be used as counters or as accumulators for aggregation.

Prototype: CARRY(varlist)

Where varlist is a list of R-variables.

Example:

```
CARRY(R1,R5-R10,R12)
```

MDCODES. The MDCODES statement changes dictionary missing data codes for input variables or assigns missing data codes for result variables. Defaults used by Recode for R- and V-variables with no dictionary missing data specification and no MDCODES specification are MD1=1.5×10^9 and MD2=1.6×10^9.

Prototype: MDCODES (varlist1)(md1,md2),(varlist2)(md1,md2), ..., (varlistn)(md1,md2)

Where:

- varlist1, varlist2, ..., varlistn are variable lists containing lists of single variables and variable ranges.
- md1 and md2 are first and second missing data codes respectively, for all variables listed. Decimal valued missing data codes must be specified with explicit decimal point. Warning: only 2 decimal places are retained for R-variables, rounding up the values accordingly, e.g. md1 specified as 9.999 is treated as 10.00.
- Either md1 or md2 may be omitted. If md1 is omitted, a comma must precede the md2 value.

Examples:

```
MDCODES V5(8,9)
```

The first missing data code for V5 will be 8; the second missing data code will be 9.

```
MDCODES (R9-R11)(,99), V7(8,9), V6(9)
```

For R9, R10 and R11, the first missing data code will be 1.5×10^9 and the second missing data code will be 99.
For V7, the first missing data code will be 8 and the second missing data code will be 9.
For V6, the first missing data code will be 9 and the second missing data code will be 1.6×10^9.

NAME. The NAME statement assigns names to R-variables or renames V-variables.

Prototype: NAME var1 'name1' ,var2 'name2', ..., varn 'name n'

Where:

- var1,var2,...,varn are V- or R-variables.
- name1, name2,...,name n are names to assign to these variables.
- The maximum number of characters per name is 24; if longer, the name is truncated to 24 characters.
- Default name for an R-variable is 'RECODED VARIABLE Rn'.
- To include an apostrophe in a name (e.g. PERSON'S), use two primes (e.g. PERSON"S).

Example:

```
NAME R1 'V5 + V6', V1 'PERSON''S STATUS'
```

4.15 Examples of Use of Recode Statements

Suppose a data file exists with the following variables:

V1	Village ID	
V2	Sex	1=male, 2=female
V4	Age	21-98, 99=not stated
V5	Education level	1=primary, 2=secondary, 3=university, 9=Not stated
V8	Income from 1st job	
V9	Income from 2nd job	
V10	Partner's income	
V21	Weight in kg (one decimal)	
V22	Height in meters (2 decimals)	
V31	Owns car?	1=yes, 2=no, 9=NS
V32	Owns TV?	
V33	Owns stereo?	
V34	Owns freezer?	
V35	Owns Micro computer?	
V41	Number of children	
V42	Age of lst child	
V43	Age of 2nd child	
V44	Age of 3rd child	
V45	Age of 4th child	

Ways to construct some possible analysis variables from this data are outlined below.

1. Total Income. If income from lst and 2nd jobs are both missing, then the total income will be missing. If only one is missing, then use this as the total.

```
          IF NVALID(V8,V9) EQ 0 THEN R101=-1 AND GO TO END
          IF NVALID(V8,V9) EQ 2 THEN R101=V8+V9 AND GO TO END
          IF MDATA(V8) THEN R101=V9 ELSE R101=V8
END       CONTINUE
          MDCODES R101(-1)

or        R101=SUM(V8,V9,MIN=1)
          IF R101 EQ 1.5 * 10 EXP 9 THEN R101=-1
          MDCODES R101(-1)
```

2. Do not use the case if total income is zero or missing.

```
IF MDATA(R101) OR R101 EQ 0 THEN REJECT
```

3. Composite income taking 3/4 of own income plus 1/4 of partner's income. If partner's income is missing, assume zero.

```
IF MDATA(V10) THEN V10=0
IF MDATA(R101) THEN R102=MD1(R102)  -
   ELSE R102=R101 * .75 + V10 * .25
NAME R102'Composite income'
MDCODES R102(99999)
```

4. Weight of respondent grouped into light (30-50), medium (51-70) and heavy (70+).

```
R103=BRAC(V21,30-50=1,50-70=2,70-200=3,ELSE=9)
```

   Note that V21 is recorded with a decimal place. To make sure that values such as 50.2 get assigned to a category, ranges in the BRAC statement should overlap. Recode works from left to right and assigns the code for the first range into which the case falls. Thus a value of 50.0 will fall in category 1 but a value 50.1 will fall into category 2. To put values of 50 in the 2nd category, use

```
R103=BRAC(V21, <50=1, <70=2, <200=3, ELSE=9)
```

   A value of 49 would fit in all 3 ranges, but Recode will use the first valid range it finds (code 1). A value of 50 will not satisfy the first range and will be assigned code 2.

5. Affluence index with values 0-5 according to the number of possessions owned.

```
R104=COUNT(1,V31-V35)
```

   If all items are coded 1 (yes), the index, R104, will take the value 5. If all are coded 2 (no) or are missing, then the index will be zero.

6. Create 3 dummy variables (coded 0/1) from the education variable.

```
DUMMY R105-R107 USING V5(1)(2)(3)
```

   The 3 result variables will take values as follows:

V5=1	R105=1, R106=0, R107=0
V5=2	R105=0, R106=1, R107=0
V5=3	R105=0, R106=0, R107=1
V5 not 1,2 or 3	R105=0, R106=0, R107=0 (default if no ELSE value given)

7. Age of youngest child. Ages of the last 4 children are stored in variables 42 to 45, the oldest child being in V42. If someone has 3 children, then the value of V44 gives the age of the youngest child; if someone has 4 or more children then we want V45. In this case, V41 (number of children) can be used as an index to select the correct variable using the SELECT function.

```
IF V41 GT 4 THEN V41=4
IF V41 EQ 0 OR MDATA(V41) THEN  R109=99 ELSE  -
   R109=SELECT (FROM=V42-V45, BY=V41)
NAME R109'Last child''s age'
MDCODES R109(99)
```

8. Weight/Height ratio as a decimal number and rounded to the nearest integer.

```
IF MDATA (V21,V22) OR V22 EQ 0 THEN R111=99 AND R112=99 -
   ELSE R111=V21/V22 AND R112=TRUNC ((V21/V22) + .5)
NAME R111'Weight/Height ratio dec', R112 'W/H rounded'
MDCODES (R111,R112)(99)
```

9. Create a single variable combining sex and educational level into 4 groups as follows:

   Females, primary education only
   Females, secondary+ education
   Males, primary education only
   Males, secondary+ education

   Method a. First reduce the codes for sex and education into contiguous codes starting from 0, storing the results temporarily in variables R901, R902.

```
R901=BRAC (V5,1=0,2=1,ELSE=9)
R902=BRAC (V6,1=0,2=1,3=1,ELSE=9)
```

   Then use the COMBINE function, making sure first that cases with spurious codes are put in a missing data category.

```
IF R901 GT 1 OR R902 GT 1 THEN R110=9 ELSE -
   R110=COMBINE R901(2),R902(2)
```

   Method b. Use IFs, setting a default value of 9 at the start.

```
R110=9
IF V5 EQ 1 AND V6 EQ 1 THEN R110=1
IF V5 EQ 1 AND V6 INLIST (2,3) THEN R110=2
IF V5 EQ 2 AND V6 EQ 1 THEN R110=3
IF V5 EQ 2 AND V6 INLIST (2,3) THEN R110=4
```

   Method c. Use the RECODE function.

```
R110=RECODE V5,V6(1/1)=1,(1/2-3)=2,(2/1)=4,(2/2-3)=5,ELSE=9
```

10. Aggregating cases with Recode. Suppose we want to analyze the data (consisting of individual level records) at the village level, for example to produce a table showing the distribution of villages by income (V8,V9) and % of people owning a car (V31) in the village. We could do this by using AGGREG to aggregate the data to the village level and then executing TABLES. Alternatively, we may use the CARRY, EOF and REJECT statements of the Recode language and use TABLES directly.

```
 1          CARRY (R901,R902,R903,R904)
 2          IF (R901 EQ 0) THEN R901=V1
 3          IF (R901 NE V1) THEN GO TO VIL
 4          IF EOF THEN GO TO VIL
 5          R902=R902+1
 6          R903=R903+V8+V9
 7          IF (V31 EQ 1) THEN R904=R904+1
 8          REJECT
 9   VIL    R101=(R904*100)/R902
10          R101=BRAC(R101,<25=1,<50=2,<75=3,<101=4)
```

```
11          R102=R903/R902
12          R102=BRAC(R102,<1000=1,<2000=2,<5000=3,ELSE=4)
13          R901=V1
14          R902=1
15          R903=V8+V9
16          IF (V31 EQ 1) THEN R904=1 ELSE R904=0
17          NAME R102'average income', R101'% owning car'
```

R901 is a work variable used to hold the current village ID; when the first case is read (R901=0), R901 is assigned the value of the village ID (V1); R902 to R904 are work variables for, respectively, the number of people in the village, the total income of the people in the village and the number of people owning cars in the village.

While the village ID stays the same, data is accumulated in variables R902 to R904 (whose values are "carried" as new cases are read). The case is then rejected (not passed to the analysis) and the next case read. When a change in village ID is encountered, the instructions at label VIL are executed: the current contents of R902, R903 and R904 are used to compute the required variables (grouped mean income and grouped % of car owners) and these variables are then passed to the analysis after first resetting the work variables to the values for the last case read (the first case for the next village). When the end of file is reached, we need to make sure that the data from the last village is used. Statement 4 achieves this.

4.16 Restrictions

1. Maximum number of R-variables is 200.
2. Maximum number of numbered tables (BRAC, RECODE, TABLE) is 20.
3. Maximum number of characters in a Recode statement excluding continuation -'s is 1024.
4. Maximum number of statement labels is approximately 60.
5. Maximum number of constants, including those in all tables, is approximately 1500.
6. Maximum number of names that may be defined in NAME statements is 70.
7. Maximum number of missing data values that may be defined in MDCODES statements is 100 and only 2 decimal places are retained for R-variables.
8. Maximum number of parenthetical nestings within a statement (i.e. parentheses within parentheses) is 20.
9. Maximum number of arithmetic operators is approximately 400.
10. Maximum number of variables with SELECT statement is 50.
11. Maximum number of IF statements is approximately 100.
12. Maximum number of function nestings (i.e. function references as function arguments) is 25.
13. Maximum number of statements is approximately 200.
14. Maximum number of labels in a BRANCH statement is 20.
15. Maximum number of CARRY variables is 100.
16. The "maximum number of variables" given in the "Restrictions" section of each analysis program write-up includes R- and V-variables used in the analysis and V-variables used in Recode but not used in the analysis. Thus, if a program has a 40-variable maximum and 40 input variables are used in the analysis, one cannot use any other input variables than those 40 in the Recode statements. R-variables defined in Recode statements but not used in the analysis need not be counted toward the "maximum number of variables".
17. Filtering takes place prior to recoding so that result variables may not be referenced in main filters.

4.17 Note

Univariate/bivariate recoding can be achieved using TABLE, IF or RECODE method. Below is a brief comparison of these methods taking into account two execution aspects.

Completeness

- TABLE...performs complete recoding. A result value is produced even when the input value is outside the table (since ELSE defaults to 99).
- RECODE allows partial recoding. If no test is true, and no ELSE value is specified, no recoding occurs.

Size of table

- Large, complete bivariate and univariate recodings are performed most efficiently by TABLE and IF...
- For a large one-to-one, univariate recoding, using one line of a rectangular table, TABLE is better than IF...

Chapter 5

Data Management and Analysis

5.1 Data Validation with IDAMS

5.1.1 Overview

Before starting analysis of data with whatever software, data normally need to be validated. Such validation typically comprises three stages:

1. Checking data completeness, i.e. verifying that all cases expected are present in the data file and that the correct records exist for each case if there are multiple records per case.
2. Checking that numeric variables have only numeric values and checking that values are valid.
3. Consistency checking between variables.

Like much other statistical software, IDAMS requires that there must be the same amount of data for each case. If the data for one case spans several records, then each case must comprise exactly the same set of records. If certain variables are not applicable to some cases, then "missing" values must none-the-less be assigned. Record merge checking capabilities in IDAMS allow for checking that each case of data has the correct set of records. This is performed by the program MERCHECK which produces a "rectangular" output file where extra/duplicate records have been deleted and cases with missing records have either been dropped or else padded with dummy records.

Checking for non-numeric values in numeric variables and the optional conversion of blank fields to user specified numeric values is performed by the BUILD program. Checking for other invalid codes is performed by the program CHECK where what are valid codes are defined on special control statements or taken from C-records in the dictionary describing the data.

If data are entered using the WinIDAMS User Interface, non-numeric characters (except empty fields) in numeric fields are not allowed. Moreover, there is a possibility of code checking during data entry and of an overall check for invalid codes in the whole data file. C-records in the dictionary are used for this purpose.

Consistency checks can be expressed in the IDAMS Recoding language and used with the CONCHECK program to list cases with inconsistencies.

Errors found in any of these steps can be corrected directly through the User Interface or by using the IDAMS program CORRECT. A typical sequence of steps for data error detection and correction with IDAMS is described in more detail below.

5.1.2 Checking Data Completeness

Step 1 Produce summary tables showing the distribution of cases amongst sampling units, geographical areas, etc. for checking against expected totals. This is particularly useful in a sample survey. For example, suppose a survey of households is done. A sample is taken by first

selecting primary sampling units (PSU) then up to 5 areas within each PSU and then interviewing households in those areas. The distribution of households by PSU and area in the data can be produced by preparing a small dictionary containing just the 2 variables: PSU and area. The table would look something like this:

			V2 AREA				
			01	02	03	04	05
		01	3	6	2		
V1	PSU	02	10	4	2	8	5
		03					
		.					
		.					

This table could be compared with the interviewers' log-book to check whether the data for all interviews taken exist in the file.

Steps 2, 3 and 4 are necessary only when cases are composed of more than one record.

Step 2 The original "raw" data records are sorted into case identification/record identification order using the SORMER program.

Step 3 The sorted raw data are checked with MERCHECK to see if they have the correct set of records for each case. The output file contains only "good" cases, i.e. ones with the correct records. Extra records and duplicate records are dropped. Cases with missing records are either dropped or padded. All cases with merge errors are listed.

Step 4 Corrections are now made for the errors detected by MERCHECK. These can be done in a variety of ways:

- Re-enter "bad" cases and merge them with the output file of MERCHECK using SORMER.
- Correct the original raw data with an editor and re-do steps 2 and 3.
- Re-enter "bad" cases, perform steps 2 and 3 on these and then merge the output from this execution of step 3 with the original output from step 3.

Whichever method is selected, MERCHECK should be re-executed on the corrected file to make sure all errors have been dealt with.

5.1.3 Checking for Non-numeric and Invalid Variable Values

Step 5 Prepare a dictionary for all variables with appropriate instructions for dealing with blank fields. Execute BUILD. An IDAMS dataset is output (Data file and Dictionary file). All unexpected non-numeric values are converted to 9's and reported in the results.

Step 6 Using TABLES, print frequency distributions of all qualitative variables and minimum, maximum and mean values for quantitative variables. This gives an initial idea of the content of the data and shows which variables have invalid codes (qualitative variables) or too large/small values (quantitative variables). It also can be compared later with a similar distributions and values obtained after cleaning to see how data validation has affected the data.

Step 7 Prepare control statements specifying the valid codes or range of values for each variable. These can be prepared ahead of time for all variables or alternatively, after step 6 for only those variables which are known to have invalid codes. Use the output dataset from step 5 as input to the CHECK program to get a list of cases with invalid values. Note that the specification of valid codes for variables can also be taken from C-records in the dictionary if these were introduced in step 5.

Step 8 Prepare corrections for errors detected at step 5 and step 7. Use the CORRECT program to update the IDAMS dataset created in step 5.
Note that corrections could also be done with the WinIDAMS User Interface if the number of cases is not too large. However using CORRECT is a less error prone method.

Perform steps 7 and 8 until no errors are reported.

5.1.4 Consistency Checking

Step 9 Prepare logical statements of the consistency checks to be performed, e.g.
PREGNANT (V32) = inapplicable if and only if SEX (V6) = Male.
Assign a "result" number to each consistency check and translate the logic into Recode statements where the result is set to 1 for an inconsistency, e.g.

```
IF V6 EQ 1 AND V32 NE 9 THEN R1001=1
IF V6 NE 1 AND V32 EQ 9 THEN R1001=1 ELSE R1001=0
```

Use the set of Recode statements with CONCHECK to print cases with errors.

Step 10 Correct cases with errors as in step 8.

Perform steps 9 and 10 until no errors are reported. The data output from the final execution of CORRECT will be ready for analysis.

5.2 Data Management/Transformation

IDAMS contains an extensive set of facilities for generating indices, derived measures, aggregations, and other transformations of the data, including alphabetic recoding. The most frequently used capabilities are provided by the Recode facility, which can perform temporary operations in all analysis programs that input an IDAMS dataset. Results of recoding can be saved as permanent variables using the TRANS program. These facilities operate on variables within one case and permit recoding of the values of one or more variables, generation of variables by combinations of variables, control of the sequence of these operations through tests of logical expressions, and a number of specialized statements and functions. The necessary new dictionary information to describe the results of the operations performed is automatically produced.

For aggregation across cases, the AGGREG program is available. AGGREG provides arithmetic sums and related measures, ranges, and counts of valid data values within groups of cases. Typical use of AGGREG involves the prior use of the SORMER program to order the Data file into the desired groups.

There are a number of circumstances in which it is necessary to combine the records from two different files, for example, data collected at different points in time. As values for variables for each new wave are received, the objective is to add them to the record containing all the previous data for the same respondent or case. The MERGE program will accomplish this, including appropriate padding with missing data where respondents are not found in the new wave. Similar examples occur when residuals or some form of scale scores are generated for each case by an analysis program and need to be included with the original data.

A somewhat different combination process occurs when data from different levels of analysis are to be combined. One illustration of this is the addition of household data to individual respondent's records. When a dataset is ordered such that all respondents in the same household are together, MERGE will provide the necessary duplicate record merge. A similar situation occurs when group summaries from AGGREG are to be added to the records for each case in each respective group.

Another dataset combination process, often also termed a merge, occurs when additional cases are to be added to a dataset. The new records must be described by the same dictionary as the original data. This type of merge may be achieved with the SORMER program.

Sub-setting functions are available as temporary operations in most IDAMS programs (by using a "filter") to select particular cases for processing. Permanent files containing subsets of IDAMS datasets (a subset of variables or a subset of cases, or both) may also be created. The SUBSET and TRANS programs are most likely to be used for such tasks, although several other programs that output datasets, such as MERGE, may also be used. Selection of cases may be done on the basis that only certain cases are logically of interest (such as only the female respondents), or it may be done on a random basis using the Recode function RAND with the TRANS program.

A display of the actual values stored in an IDAMS dataset is often of substantial help for checking the results from data modification steps and indeed at any other stages. The LIST program is available for this purpose, and allows complete listings of a selection of specific cases and variables. The selection or filtering of cases for display may be done using combinations of several variables in logical expressions; an example would be

a selection of only records for unmarried women between 21 and 25 years of age. Numeric and alphabetic variables from a dataset as well as variables constructed with Recode statements can be listed. The User Interface also has an option to print the data in a table format.

5.3 Data Analysis

The paramount consideration for the user in selecting analysis programs is whether the appropriate statistical functions are provided. Guidance on such matters is well beyond the scope of this manual. A summary of the functions of each IDAMS analysis program can be found in the Introduction. More details are given in the individual program write-ups. The formulas used for computing the statistics in each program, and references are given in relevant chapters of the part "Statistical Formulas and Bibliographic References".

5.4 Example of a Small Task to be Performed with IDAMS

Suppose that an IDAMS dataset contains responses to a survey questionnaire and includes the following variables:

V11 gives the sex of the respondent according to the following code:

1. Male 2. Female 9. Not ascertained

V12 is the respondent's income in dollars (99999 = not ascertained).

V13 through V16 are attitudinal measures on different issues. The variables are each coded to reflect the feelings of the respondent as follows:

1. Very positive 2. Positive 3. Neutral 4. Negative 5. Very negative 8. Don't know
9. Not ascertained 0. The question is irrelevant for this respondent

Suppose that only a grouping or recoding of income levels is needed of the following kind:

New code	Meaning
1	Income in the range \$0 to \$9999
2	Income in the range \$10,000 to \$29,999
3	Income \$30,000 and over
9	Refused, Not ascertained, Don't know

Cross-tabulations are desired between the recoded version of the income variable, V12, and each of the attitudinal variables, V13 to V16. Only the female respondents are to be selected for this analysis.

An IDAMS "setup" containing the necessary control statements to perform this work is shown below. The numbers in parentheses on the left identify each control statement and link it to the subsequent explanation.

```
 (1)  $RUN TABLES
 (2)  $FILES
 (3)  DICTIN = ECON.DIC
 (4)  DATAIN = ECON.DAT
 (5)  $RECODE
 (6)      R101=BRAC(V12,0-9999=1,10000-29999=2,30000-99998=3, -
 (7)           ELSE=9)
 (8)      NAME R101 'GROUPED INCOME'
 (9)  $SETUP
(10)  INCLUDE V11=2
(11)  EXAMPLE OF TABLES USING ECONOMIC DATA
(12)  *
(13)  TABLES
(14)  ROWVARS=(R101,V13-V16)
(15)  ROWVAR=R101  COLVARS=(V13-V16) CELLS=(FREQS,ROWPCT)  STATS=CHI
```

Briefly, this is what each statement does:

(1) "$RUN TABLES" is an IDAMS command specifying that the TABLES program is to be executed.

(2) This statement signals the start of file definitions for the execution.

(3)&(4) The IDAMS dataset is stored in two separate files. One contains the dictionary, the other the data.

(5) This statement signals that transformations of the data are required. The statements following this are the specific commands to the Recode facility.

(6)(7) These two lines (an original and a continuation) form a statement to the Recode facility indicating the desired grouping for the income variable, V12, following the scheme outlined earlier. The result of the BRAC function is stored as result variable R101.

(8) This statement assigns name to the variable R101.

(9) "$SETUP" is a command which indicates the end of Recode statements and that the TABLES program control statements follow.

(10) This is a "filter" which states that the only data cases to be used are those where variable V11 has the code value 2, for females.

(11) This is a label, which contains the text to be used to title the results.

(12) This line specifies the main parameters. Since only the asterisk is given, all the default options for the parameters are chosen for the current execution.

(13) The word TABLES is supplied here to separate the preceding global information for the entire execution from the specifications for individual tables that follow.

(14) This statement requests univariate frequency distributions for 5 variables.

(15) Now bivariate (2-way) tables are requested. The cells are to contain the counts (frequencies) and row percentages; a Chi-square statistic will be printed for each table. The 2 lists of variables following the keywords ROWVAR and COLVARS specify the variables that will be used for the rows and columns of the tables respectively. Four tables will be produced: R101 (grouped income) by V13, V14, V15 and V16).

Part II

Working with WinIDAMS

Chapter 6

Installation

6.1 System Requirements

- The WinIDAMS software is available for 32-bit versions of Windows operating systems (Windows 95, 98, NT 4.0, 2000 and XP)
- A Pentium II or faster processor and 64 megabytes RAM are recommended.
- On all systems, you should have about 11 megabytes of free disk space before attempting to install the WinIDAMS software in each language.

6.2 Installation Procedure

- The release 1.3 of WinIDAMS is stored on CD in a self-extracting file

```
WinIDAMS\English\Install\WIDAMSR13E.EXE      : English version
WinIDAMS\French\Install\WIDAMSR13F.EXE       : French version
WinIDAMS\Portuguese\Install\WIDAMSR13P.EXE   : Portuguese version
WinIDAMS\Spanish\Install\WIDAMSR13S.EXE      : Spanish version
```

  or in equivalent downloaded file.

- To install the English version:
  1. Select `WIDAMSR13E.EXE` with Windows explorer.
  2. Double-click on this file and follow the prompts.
  3. At the end of the installation procedure, a dialog box appears asking: "Do you wish to install HTML Help 1.3 update now?". It is recommended to answer YES.
- The installation procedure creates two items in the Program Manager/Start menu, one for executing WinIDAMS and one for uninstalling WinIDAMS. It also creates an icon on the desktop which is a link/shortcut to WinIDAMS.

6.3 Testing the Installation

A Setup file containing instructions for executing 4 data management programs (CHECK, CONCHECK, TRANS and AGGREG) and 6 data analysis programs (TABLES, REGRESSN, MCA, SEARCH, TYPOL and RANK) is copied into the Work folder during the installation. To execute it:

- Start WinIDAMS by a double-click on its icon.

- You will see the WinIDAMS main window with a default application displayed in the left pane. Open the Setups folder. There is the demo.set file with instructions for execution of the 10 programs.
- By double-click, the file opens in the Setup window. Execute it from this window. Results of the execution are sent to the file idams.lst which is immediately opened in the Results window.
- The distributed version of the results is provided in the file demo.lst in the Results folder.
- Compare the two versions of the results.

6.4 Folders and Files Created During Installation

6.4.1 WinIDAMS Folders

The full path name of the WinIDAMS System folder is given on the "Select Destination Directory" of the installation wizard and the following folders are created during the installation (see "Files and Folders" chapter for details):

English version	French version
<WinIDAMS13-EN>\appl	<WinIDAMS13-FR>\appl
<WinIDAMS13-EN>\data	<WinIDAMS13-FR>\data
<WinIDAMS13-EN>\temp	<WinIDAMS13-FR>\temp
<WinIDAMS13-EN>\trans	<WinIDAMS13-FR>\trans
<WinIDAMS13-EN>\work	<WinIDAMS13-FR>\work

Portuguese version	Spanish version
<WinIDAMS13-PT>\appl	<WinIDAMS13-SP>\appl
<WinIDAMS13-PT>\data	<WinIDAMS13-SP>\data
<WinIDAMS13-PT>\temp	<WinIDAMS13-SP>\temp
<WinIDAMS13-PT>\trans	<WinIDAMS13-SP>\trans
<WinIDAMS13-PT>\work	<WinIDAMS13-SP>\work

6.4.2 Files Installed

System files in the System folder

(\WinIDAMS13-EN, \WinIDAMS13-FR, \WinIDAMS13-PT, \WinIDAMS13-SP)

WinIDAMS.exe	Main executable file for the WinIDAMS User Interface
Ter32.dll	|
Hts32.dll	| Dlls used by WinIDAMS User Interface
unesys.exe	Executable file used for processing setups
Idame.mst	Master file of the text data base for IDAMS programs
Idame.xrf	Cross reference file of the text data base for IDAMS programs
idams.def	Definition of the mapping between ddnames and file names
Graph32.exe	GraphID executable file
graphid.ini	Ini file used by GraphID for storing colours, fonts and co-ordinates
Idtml32.exe	TimeSID executable file
idaddto32.dll	Dll used by GraphID and TimeSID
IDAMSC_DLL.dll	Dll used by TimeSID
Idams.chm	WinIDAMS Manual help file
<pgmname>.pro	Prototypes for IDAMS programs

Dictionary and data files used for examples in the Data folder

(\WinIDAMS13-EN\data, \WinIDAMS13-FR\data, \WinIDAMS13-PT\data, \WinIDAMS13-SP\data)

```
educ.dic
educ.dat
rucm.dic
rucm.dat
watertim.dic
watertim.dat
data.csv
tab.mat
```

Demonstration setup and result files in the Work folder

(\WinIDAMS13-EN\work, \WinIDAMS13-FR\work, \WinIDAMS13-PT\work, \WinIDAMS13-SP\work)

```
demo.set
demo.lst
```

6.5 Uninstallation

An uninstaller program is created during the installation procedure. The user can execute the uninstaller either by clicking on WinIDAMS13-EN/Uninstall WinIDAMS13-EN in the Program Manager/Start menu or by deleting the "WinIDAMS Release 1.3, English version, July 2004" entry in the Add/Remove Programs Control Panel applet. This uninstaller deletes the content of the WinIDAMS folder selected during the installation process. It does not delete folders if they are not empty.

Chapter 7

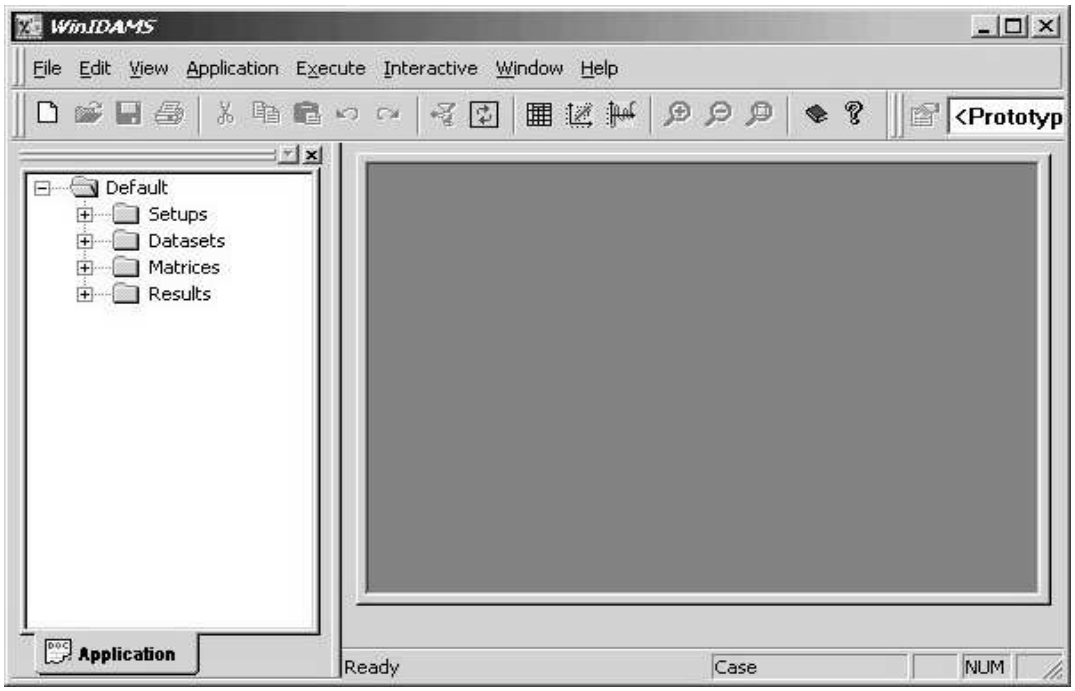
Getting Started

7.1 Overview of Steps to be Performed with WinIDAMS

In this example, an IDAMS dictionary for the description of data collected by a questionnaire is prepared and data for a few respondents are entered. A set of IDAMS control statements (a "setup") is then prepared and used to produce frequency distributions of Age, Sex and Education (number of years) bracketed into 4 groups. The steps below are followed:

1. Create an application environment.
2. Prepare and store an IDAMS dictionary describing the variables in the data.
3. Enter the data (this step would be eliminated if the data were prepared outside WinIDAMS).
4. Prepare and store a "setup" of instructions specifying what is to be done with the data.
5. Execute the IDAMS program as given in the setup.
6. Review the results and modify the setup if necessary; then repeat from step 4.
7. Print the results.

To get started, first launch WinIDAMS. You will see the WinIDAMS Main window.

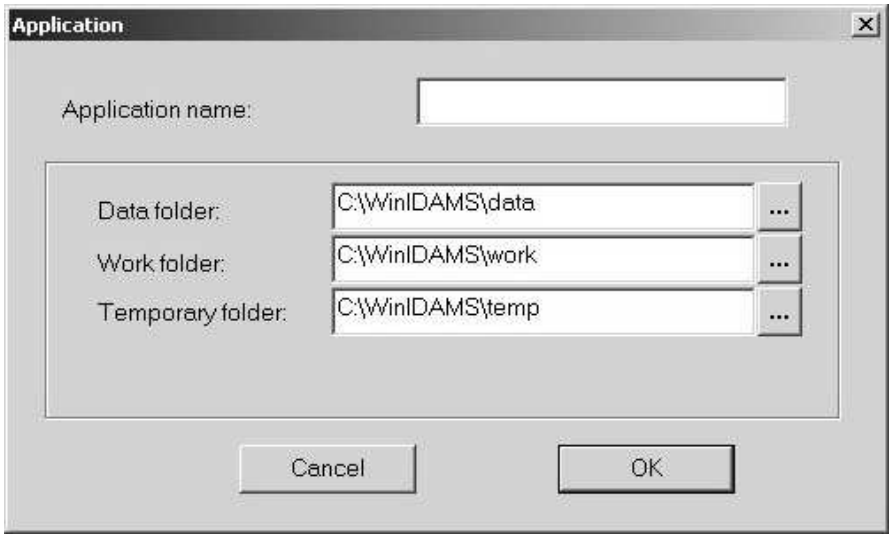


7.2 Create an Application Environment

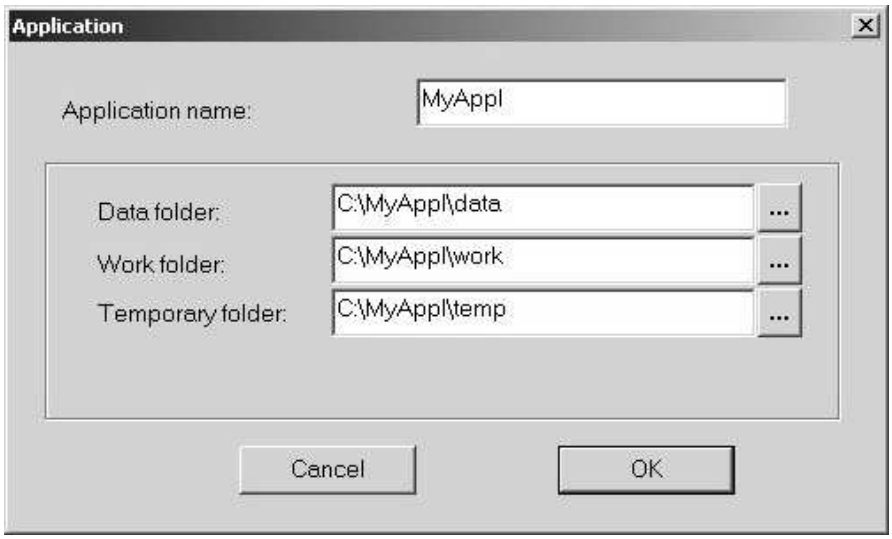
The application environment allows you to predefine full paths for three folders. All input/output files will be opened/created by default in one of these folders. This saves you from having to enter the full folder path.

- The Data and Dictionary files: in the Data folder.
- The Setup and Results files: in the Work folder.
- The temporary files: in the Temporary folder.

Click on Application in the menu bar and then on New. You now see the following dialogue:



We will create a new application with the name "MyAppl" and with application folders `C:\MyAppl\data`, `C:\MyAppl\work` and `C:\MyAppl\temp` by entering these names in the corresponding text-boxes.



For each application folder entered which does not exist, you will see a dialogue like this:

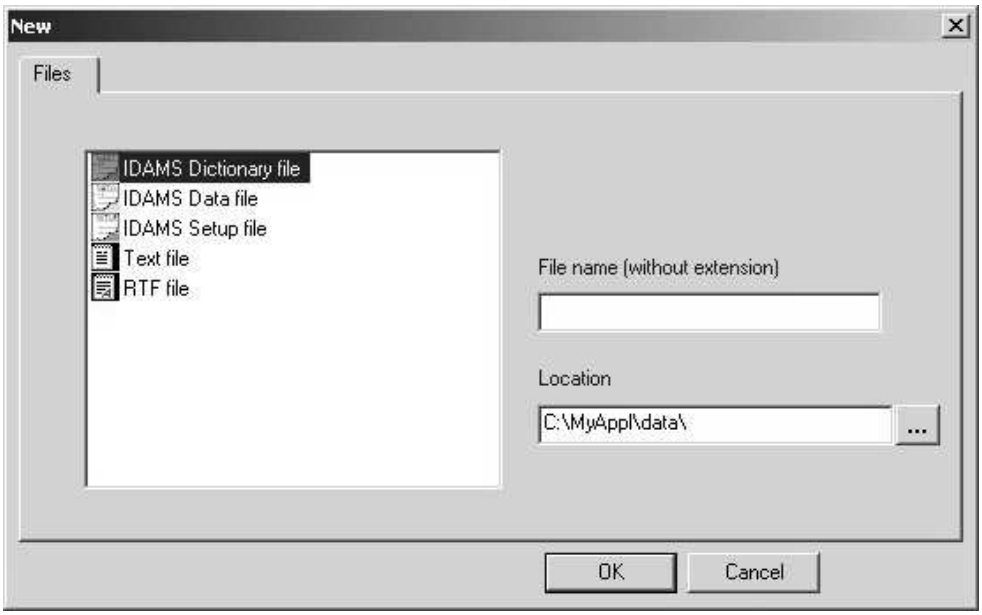
Click on Yes for each new folder and then click on OK. Now you see the WinIDAMS Main window again.

7.3 Prepare the Dictionary

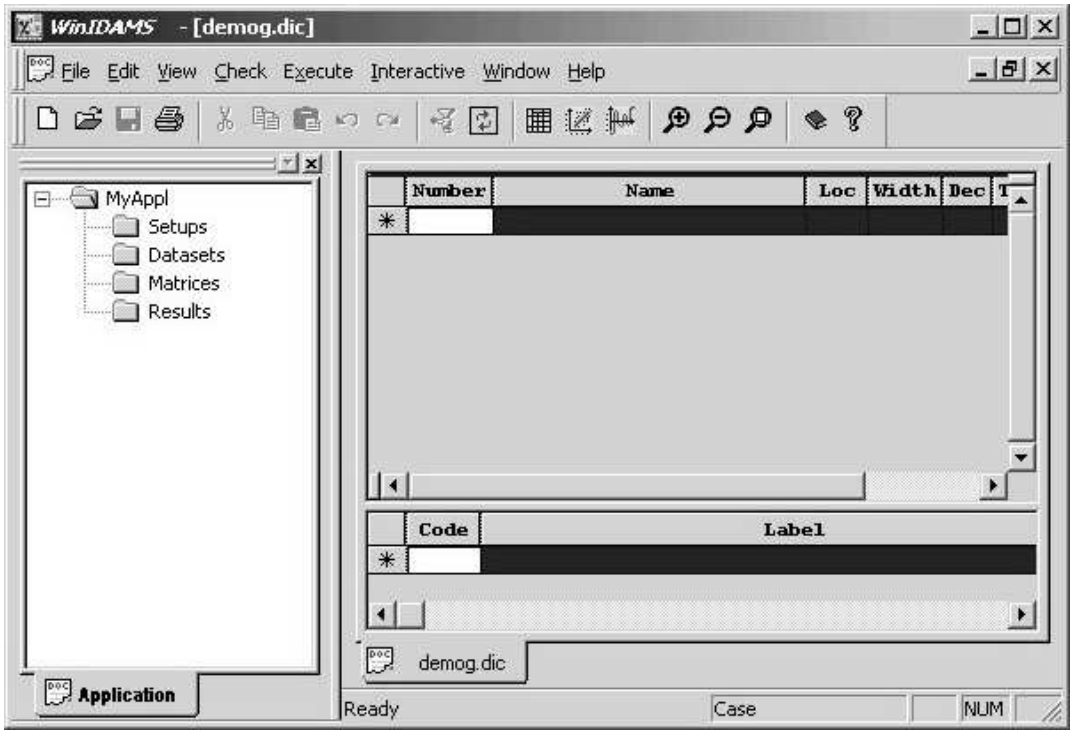
We will create a dictionary to describe data records containing the following variables:

Number	Name	Width	Missing Data code
1	Identification	3	
2	Age	2	
3	Sex	1	9
	1 Male		
	2 Female		
	9 MD		
4	Education	2	

- Press Ctrl/N or click on File/New. These commands open the New document dialogue:



- The dialogue displays the list of document types used in WinIDAMS. Choose “IDAMS Dictionary file”, already selected by default.
- Click in the File name field and enter the name “demog”. Then click OK. Note that extension .dic is added automatically to the file name.
- You now see:
  - the Application window;
  - a 2 pane window for entering variable descriptions and optional associated codes and labels. The full Dictionary file name “demog.dic” is displayed in the tab.



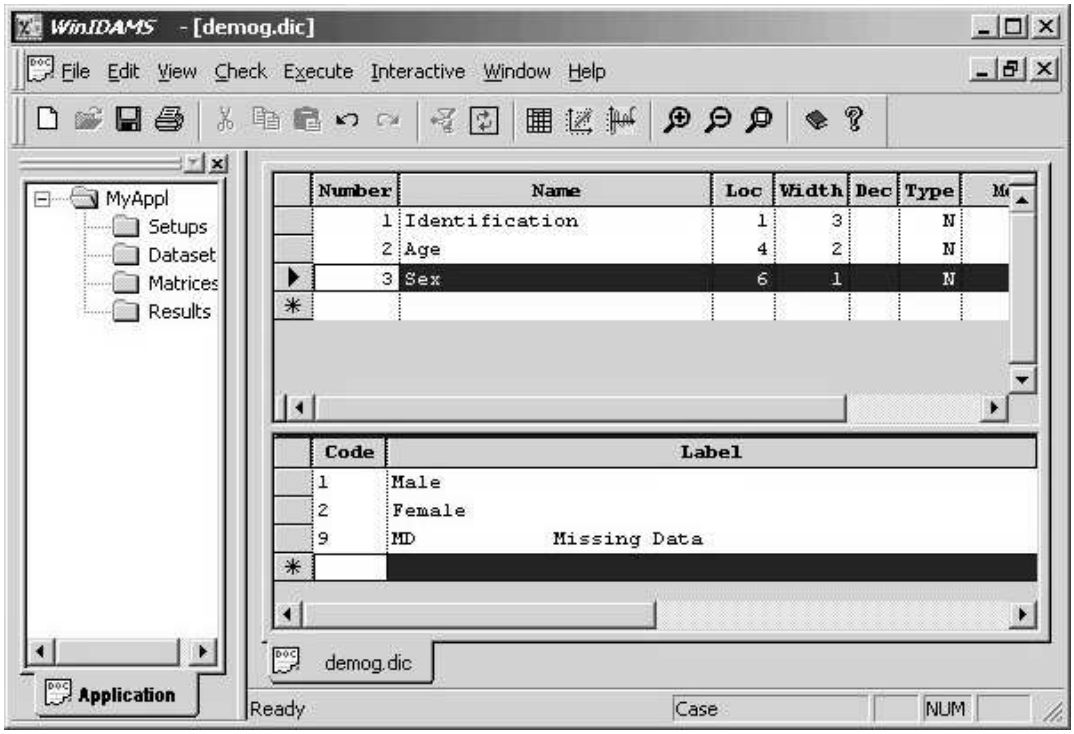
- Click on the first cell in the row of the pane for describing variables and enter the first variable number. As soon as you begin to enter information in the row marked with an asterisk, a new row is created just after the current row and the row you are editing displays a pencil in the row header. Pressing Enter or Tab you move to the next field. Now enter variable name and width. Skip the rest of fields by pressing Enter or Tab and accept the description by pressing Enter or Tab on the last field. Note that the default location is provided by WinIDAMS when variable description row has been accepted.

- When you press Enter or Tab on the last field, the pencil disappears which means that the row has been accepted after some rudimentary checking of the fields. The current field is now the first field of the next row (marked with an asterisk) and you can enter the description for the 2nd variable, Age. Do the same for variable 3, Sex, but give this variable an MD1 (missing data) code of 9 (the non-response code).

- After accepting the description of variable 3, the first field (variable number) of the row with an asterisk becomes the current field. Click on any field of the row just entered (variable 3, Sex) to make it the current row.

- Switch to the pane for codes and their labels by clicking on the code field in the first row. Note that this pane is synchronized with the variable selected in the pane for describing variables.

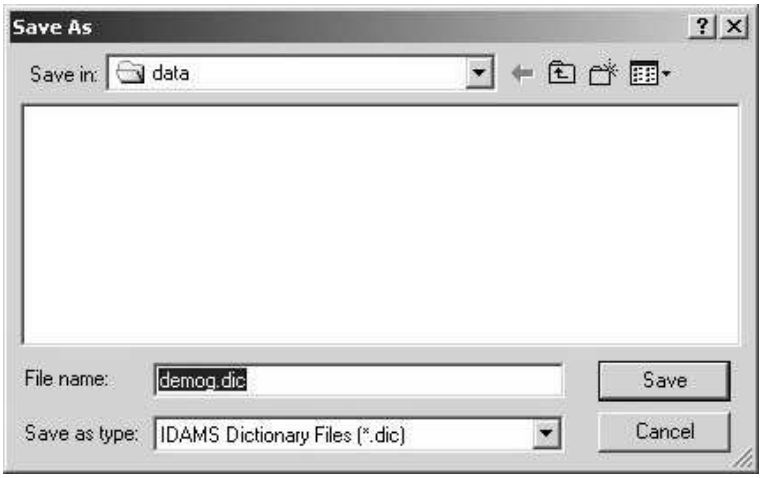
- Enter 1 in the code field. Again, as soon as you begin to enter code label, a new row with an asterisk is created just after the current row and the row you are editing displays a pencil. Press Enter to move to the next field, enter Male in the label field. Press Enter. The current field is now the code field of the next row and you can enter code 2 with label Female and similarly for code 9.



- Go back to the variable description pane by clicking on the variable number field of the row with an asterisk. Enter the information for variable 4.

  To delete rows, click at the side of the row and select Cut from the Edit menu.

- Save the dictionary by clicking on File/Save As, and accepting the Dictionary file name "demog.dic".

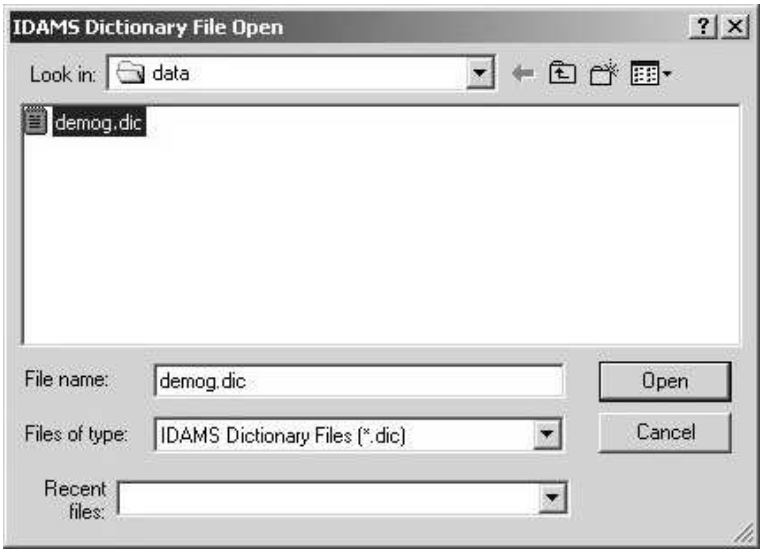


7.4 Enter Data

- Press Ctrl/N or click on File/New. The same New document dialogue as we have seen above for the dictionary is displayed.

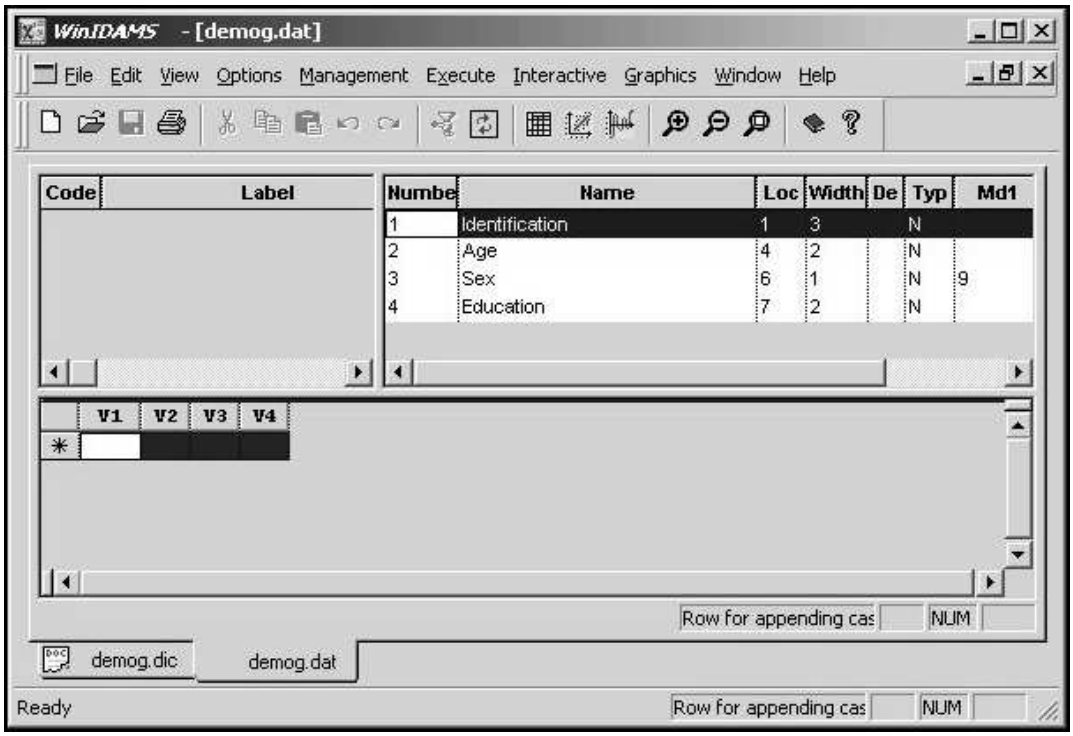
- Select the "IDAMS Data file" item from the list and enter the name of the Data file. By convention, it is better to use the same name for the Data file and the Dictionary file which describes the data. Only the file extension changes, ".dic" for the Dictionary file and ".dat" for the Data file. The dictionary and data make up an IDAMS dataset. Enter "demog" as file name and click on OK.

- A File Open dialogue now displays the dictionaries which exist for the active application and asks you to select the dictionary which describes the data. Select "demog.dic" and click Open.



- A window with three panes now appears. You enter data only in the bottom pane. The 2 other panes are synchronized for displaying the current variable description and the code labels if any. The full Data file name "demog.dat" (extension .dat is added automatically) is displayed in the tab.

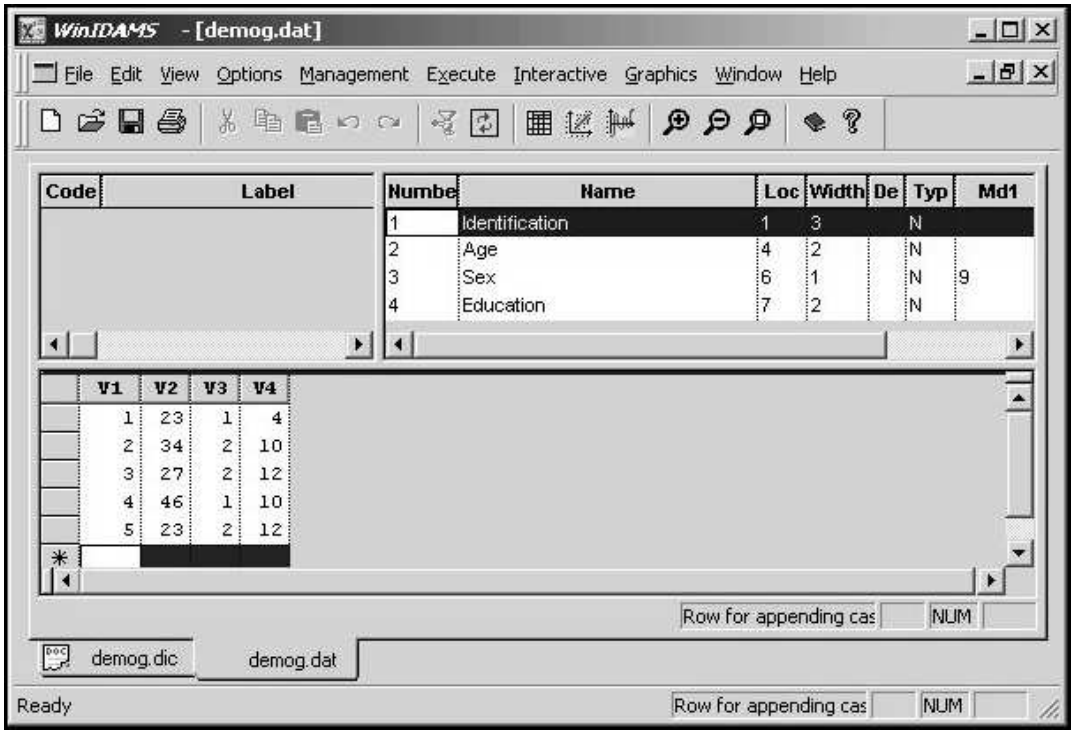
  Note that in illustrations presented below the Application window has been closed.


WinIDAMS - [demog.dat]
File Edit View Options Management Execute Interactive Graphics Window Help
Code Label
Numbe Name Loc Width De Typ Md1
1 Identification 1 3 N
2 Age 4 2 N
3 Sex 6 1 N 9
4 Education 7 2 N
V1 V2 V3 V4
Row for appending cas NUM
demog.dic demog.dat
Ready
Row for appending cas NUM

- Click on the first field of the row with an asterisk and type the first line of data as given below, pressing the Enter key after entering each data value. As soon as you begin to enter data, a new row is created just after the current row and the current row header displays a pencil which means that you are editing this row.

- After entering the value for the last variable V4 and pressing Enter, the first field of the next row becomes the current field.

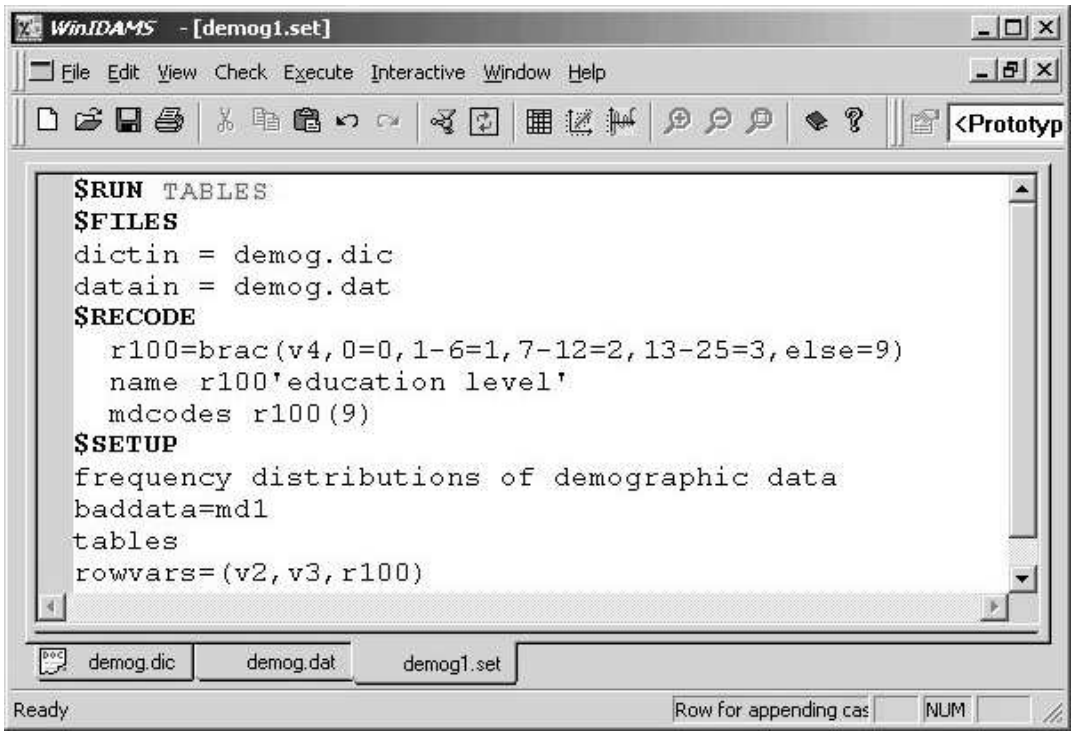
- Enter the data for the 5 cases given below.



- Click on File/Save to save the data in the file "demog.dat".

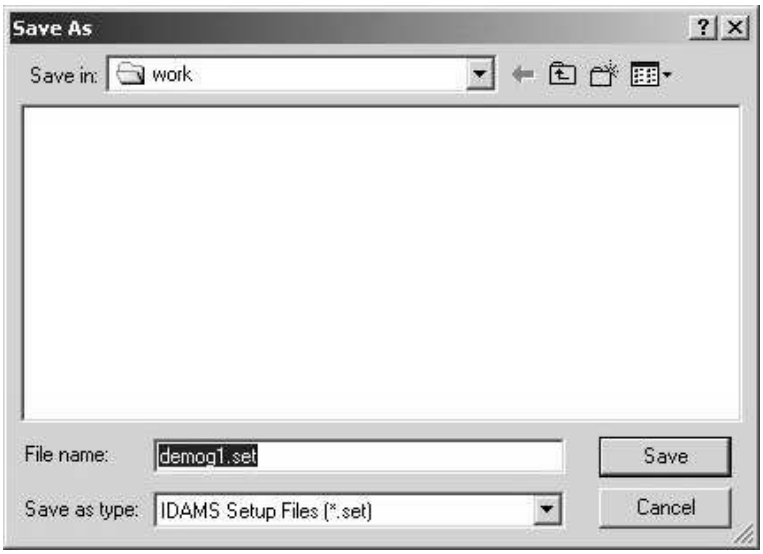
7.5 Prepare the Setup

- Press Ctrl/N or click on File/New.
- Select the "IDAMS Setup file" item from the list and enter a name, e.g. "demog1" for the Setup file. Click OK. Note that extension .set is added automatically to the file name and the full file name "demog1.set" is displayed in the tab.
- You will now see an empty window for entering the setup. Type the following:



The $RUN identifies the desired IDAMS program; following the $FILES command, the Data file and associated Dictionary file are specified; the $RECODE command followed by Recode statements (here the recoding is used to bracket years of education into 4 groups); the $SETUP command followed by the parameters for the task (in this case requesting univariate frequency distributions) are given (according to the rules for the TABLES program).

- Click on File/Save and save the setup in the file "demog1.set".

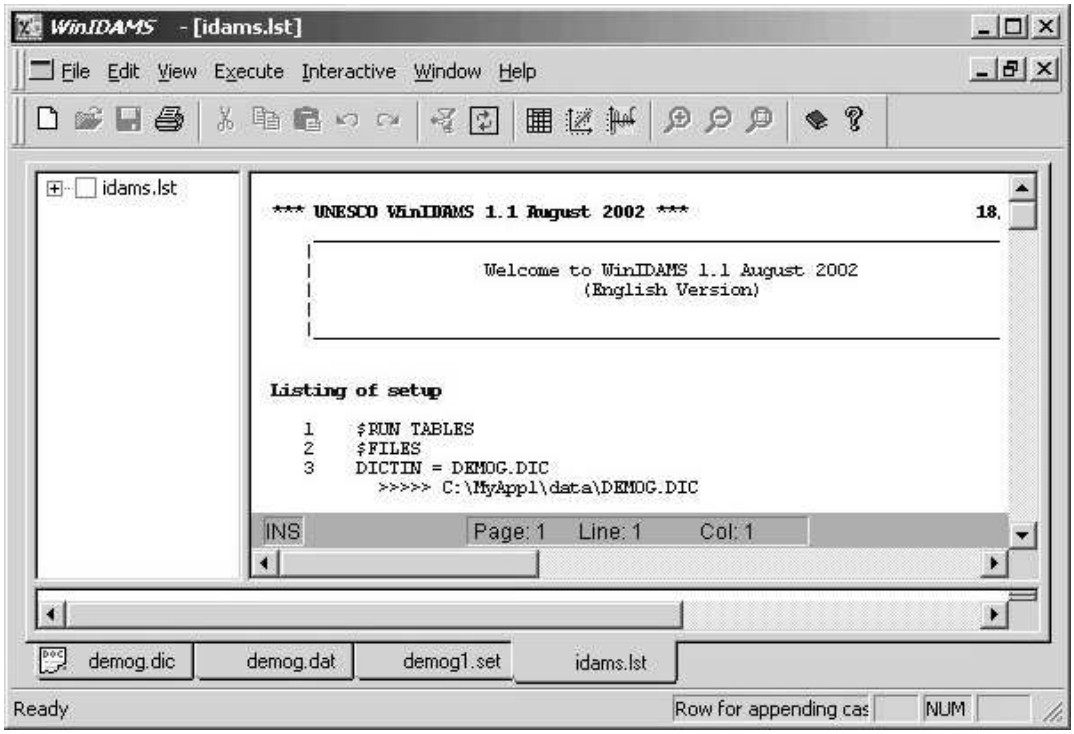


7.6 Execute the Setup

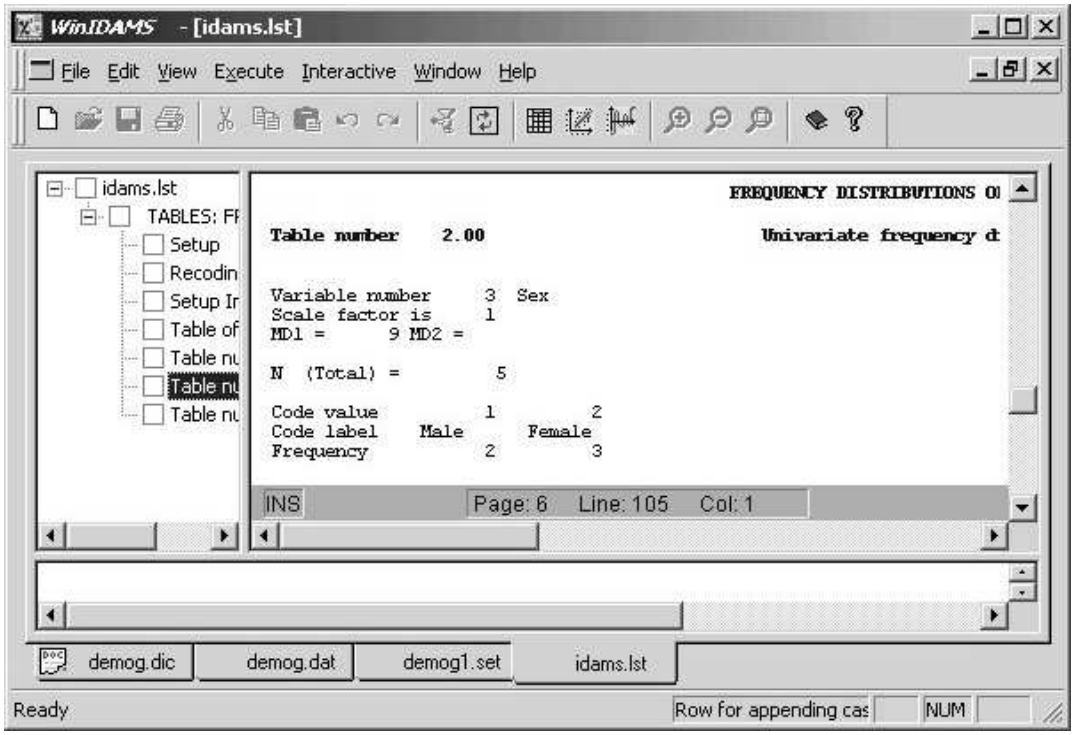
- From inside the Setup window, click on Execute/Current Setup. The current setup is saved in a temporary file and executed. A dialogue appears during the execution and disappears if the execution is successful.
- The results are, by default, written into the file "idams.lst". It can be changed by adding a PRINT line under $FILES for giving the name of Results file, e.g. "print=a:demog1.lst" to store the results in a file on diskette.

7.7 Review Results and Modify the Setup

- The Results file is loaded automatically when the execution is finished.



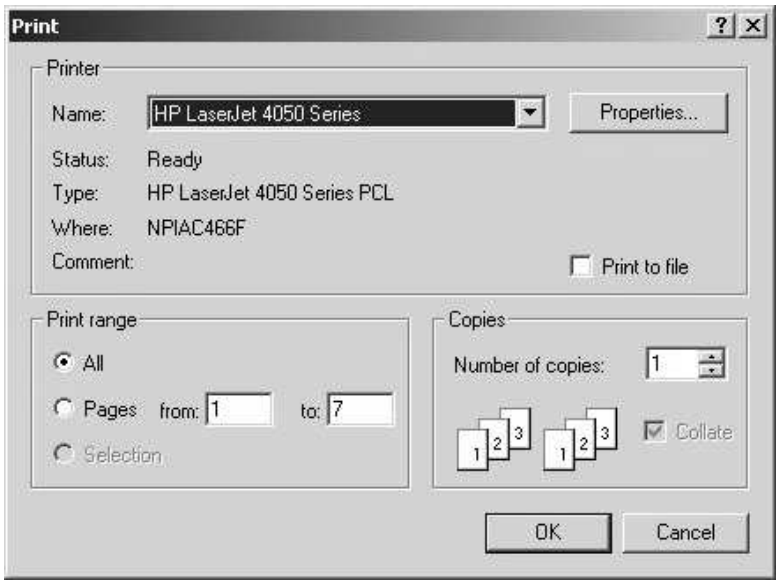
- The table of contents provided in the left pane allows quick location of parts of the results. Open it by clicking "idams.lst" and pushing button with an asterisk on the numeric pad. Then, click on the element you want to see.



- If you want to change something in the setup while reviewing the results, then click on the tab "demog1.set" and make the required modifications. Press Ctrl/E to execute.

7.8 Print the Results

- Select File/Print.



- Select the pages that you wish to print and click on OK.

Chapter 8

Files and Folders

8.1 Files in WinIDAMS

User files

They are created by the user with the help of tools provided by the WinIDAMS User Interface, or they are produced by an IDAMS procedure as a final result or as output for further processing. All user files in IDAMS are ASCII text files. Tabulation characters are allowed; they are automatically converted to the correct number of blanks. Standard filename extensions are used by the Interface for recognizing the file type.

- ***Data file (*.dat).*** Any data file can be input to IDAMS programs providing that each case is contained in an equal number of fixed format records. However, if a data file is used by the WinIDAMS User Interface, then there can only be one record per case.

  Records can be of variable length with a maximum of 4096 characters per case. If the first record in the file is not the longest, then the maximum record length (RECL) must be provided on the corresponding file specifications. Data files produced by IDAMS programs have fixed length records with no tabulation characters. There is generally no limit to the number of cases that can be input to an IDAMS program.

- ***Dictionary file (*.dic).*** The dictionary is used to describe the variables in the data. It may, at minimum, describe just the variables being used for a particular program execution, but it can also describe all the variables in each data record. The record length is variable but the maximum length is 80. If a dictionary is output by an IDAMS program, then the record length is fixed (80 characters) with no tabulation characters.

  The dictionary can be prepared, without knowing its internal format, in the Dictionary window of the User Interface. Alternatively, it can be prepared using the General Editor and following the format given in "Data in IDAMS" chapter.

- ***Matrix file (*.mat).*** IDAMS matrices for storing various statistics have fixed length (80 characters) records with no tabulation characters.

- ***Setup file (*.set).*** This file is used to store IDAMS commands, file specifications, program control statements and Recode statements (if any). The Setup file can be prepared in the Setup window of the User Interface. The record length is variable although the maximum is 255 characters.

- ***Results file (*.lst).*** IDAMS normally writes the results into a file. The contents of this file can then be reviewed before actually printing.

Note: In order to facilitate the work with WinIDAMS, it is advisable to use a common name for Data and Dictionary files, and also a common name for Setup and Results files.

The user files are specified in the Setup file following the $FILES command (see "The IDAMS Setup File" chapter for detailed description).

System files

System files are normally not accessed directly by the user. They are created during the installation process (permanent System files), during application customization (Application files) or during the execution of WinIDAMS procedures (temporary work files).

- ***Permanent System files.*** These include the executable program files, dll files, system parameter files, file with the on-line Manual (in HTML Help format), and setup prototype files.
- ***System control files.***
  - *Idams.def :* default file definitions providing connection between logical and physical filenames for user files and temporary work files.
  - *<application_name>.app :* one file per application containing paths of Data folder, Work folder and Temporary folder.
  - *lastapp.ini :* file containing the name of the last application used.
  - *graphid.ini :* configuration settings for the GraphID component.
  - *tml.ini :* configuration settings for the TimeSID component.
- ***Temporary work files.*** They need not concern the user since they are defined and removed automatically. They have filename extensions .tmp and .tra.

8.2 Folders in WinIDAMS

Files used in WinIDAMS are stored in the following folders:

- System files in the *System folder*,
- Application files in the *Application folder*,
- Data, Dictionary and Matrix files in the *Data folder*,
- Setup files and Results files in the *Work folder*, and
- temporary work files in the *Temporary folder* and *Transposed folder*.

Five folders, mandatory for the default application, should always be present under the `<system_dir>` folder. They are defined and created first during the installation process, Then, when WinIDAMS is started and any of the folders is missing, it is automatically recreated.

Application folder	`<system_dir>\appl`
Data folder	`<system_dir>\data`
Temporary folder	`<system_dir>\temp`
Transposed folder	`<system_dir>\trans`
Work folder	`<system_dir>\work`

where `<system_dir>` is the name of the System folder fixed during the installation.

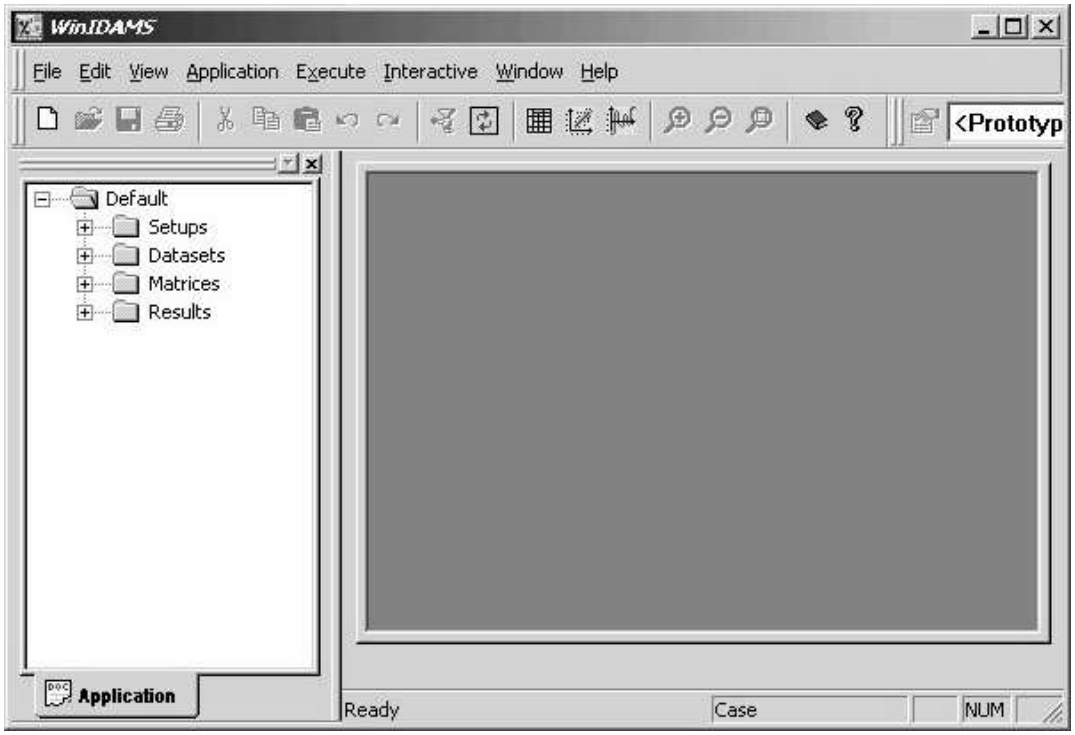
For more details on how IDAMS programs use the paths defined in the application, see section "Customization of the Environment for an Application" in the "User Interface" chapter.

Chapter 9

User Interface

9.1 General Concept

The WinIDAMS User Interface is a multiple document interface. It can display and allow to work simultaneously with different types of documents such as Dictionary, Data, Setup, Results and any Text document in separate windows. Moreover, it provides access to execution of IDAMS setups and to components for interactive data analysis, namely: Multidimensional Tables, Graphical Exploration of Data and Time Series Analysis from any document window. The WinIDAMS Main window contains:



- ***the menu bar*** to open drop-down menus with WinIDAMS commands or options,
- ***the toolbar*** to choose commands quickly,
- ***the status bar*** to display information about the active document or highlighted command/option,
- ***the Application window,*** docked on the left side, to display the active application name, and folders and documents for this application,
- ***the document windows*** to display different WinIDAMS documents.

The menu bar and the toolbar have fixed, document dependent contents. The common menus are described below while document type dependent menus are described in relevant sections.

9.2 Menus Common to All WinIDAMS Windows

The main menu bar contains always the seven following menus: File, Edit, View, Execute, Interactive, Window and Help.

File

New	Calls the dialogue box to select the type of document to be created, and to provide its name and location.
Open	After choosing the type of document, calls the dialogue box to select the document to be opened.
Close	Closes the active window.
Save	Saves the document displayed in the active window.
Save As	Calls the dialogue box to save the document in the active window.
Print Setup	Calls the dialogue box for modifying printing and printer options.
Print Preview	Displays the active document as it will look when printed.
Print	Calls the dialogue box for printing the contents of the document displayed in the active pane/window. Note that hidden parts of the document are not printed.
Exit	Terminates the WinIDAMS session.

The menu can also contain the list of up to 7 recently opened documents, i.e. documents used in previous WinIDAMS sessions.

Edit

The availability and sometimes the title of some commands in this menu may be different in different windows.

Undo	Cansels the last action.
Redo	Does again the last canceled action.
Cut	Moves the selection to the Clipboard.
Copy	Copies the selection to the Clipboard.
Paste	Copies the Clipboard content to the place where the cursor is positioned.
Find	Starts the Windows searching mechanism.
Replace	Starts the Windows replacing mechanism.
Find again/next	Looks for the next appearance of the character string displayed in the Find dialogue box.

Note that in the Results and Text windows, the search/replace actions are activated by the Search, Search Forward, Search Backward and Replace commands.

View

Toolbar	Displays/hides toolbar.
Status Bar	Displays/hides status bar.
Application	Displays/hides the Application window.
Show Full Screen	Displays the active window in full screen. Click the Close Full Screen icon in the left-top corner or press Esc to go back to the previous screen.

Execute

With exception of the Setup window, the menu has only one command, Select Setup, to select a file with the setup to be executed.

Interactive

Through this menu, three components for interactive analysis can be accessed, namely:

Multidimensional Tables
Graphical Exploration of Data
Time Series Analysis

See relevant chapters for a detailed description of each component.

Window

The menu contains the list of opened windows and standard Windows commands for arranging them.

Help

WinIDAMS Manual	Provides access to the WinIDAMS Reference Manual.
About WinIDAMS	Displays information about the version and copyright of WinIDAMS and a link for accessing the IDAMS Web page at UNESCO Headquarters.

9.3 Customization of the Environment for an Application

Names of Data folder, Work folder and Temporary folder can be defined by the user and saved in an Application file with the application name as filename. The name of the last application used is stored by the system and the settings defined for this application are loaded at the beginning of the following session. These settings can be changed any time during the working session by selecting/creating and activating another application.

Since at least one Application file is necessary for the use of WinIDAMS, a standard application called "Default" is provided and will be activated when you start WinIDAMS for the first time after installation. Defined default settings are the following:

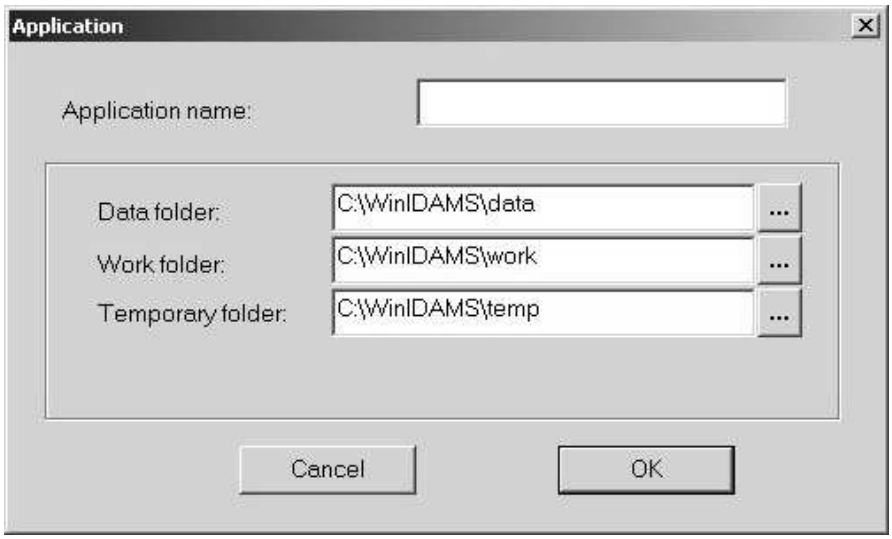
Data folder	`<system_dir>\data`
Work folder	`<system_dir>\work`
Temporary folder	`<system_dir>\temp`

where `<system_dir>` is the System folder name fixed during the installation. This application (stored in the file Default.app) should neither be deleted nor modified by the user.

Application files (except Default.app) can be created, modified or deleted by the user through the **Application** menu in the WinIDAMS Main window. It contains the following commands:

New	Calls the dialogue box for creating a new application.
Open	Calls the dialogue box to select the file containing details of the application to be opened.
Display	Calls the dialogue box to select the application file and displays the application settings.
Close	Closes the active application and opens the Default application.
Refresh	Recreates the current application tree.

Creating a new application. Selection of the menu command Application/New provides a dialogue box for entering the name of a new application as well as names of Data, Work and Temporary folders. Except the application name field which is empty, all the other fields contain default values taken from the Default application. You can type in the pathname directly or select it moving the highlight to the required name in the displayed tree of folders.



Press OK button to save the application. Pressing Cancel cancels the creation of a new application and returns to the WinIDAMS Main window with the settings displayed previously.

Opening an application. The menu command Application/Open calls the dialogue box to select an application file to be opened and provides a list of existing applications in the Application folder. Clicking the required file name activates the settings for this application.

Modifying an application. To modify an application, first open it and then change the values in the same way as for creating a new application.

Displaying the settings for an application. Use the menu command Application/Display to call the dialogue box and click the required file name.

To display settings for the active application, double-click its name in the Application window.

Deleting an application. It can be done by deleting the corresponding file. Use the menu command Application/Open to get a list of Application files, select the file to delete and use the right button to access the Windows Delete command. The file Default.app should not be deleted.

Resetting WinIDAMS defaults. To replace the displayed application by the default application you can either close it using the menu command Application/Close, or select and open the Default.app file.

Closing an active application. Use the menu command Application/Close. The default application becomes active.

IDAMS programs use the paths defined in the application to prefix any filename not beginning with "`<drive>:\...`" or with "`\...`"

- The Data folder path is prefixed to all filenames in statements with ddnames DICT..., DATA..., or FTnn referring to matrices.
- The Work folder path is prefixed to filenames in statements with ddnames PRINT or FT06.
- The Temporary folder path is prefixed to names of temporary files.

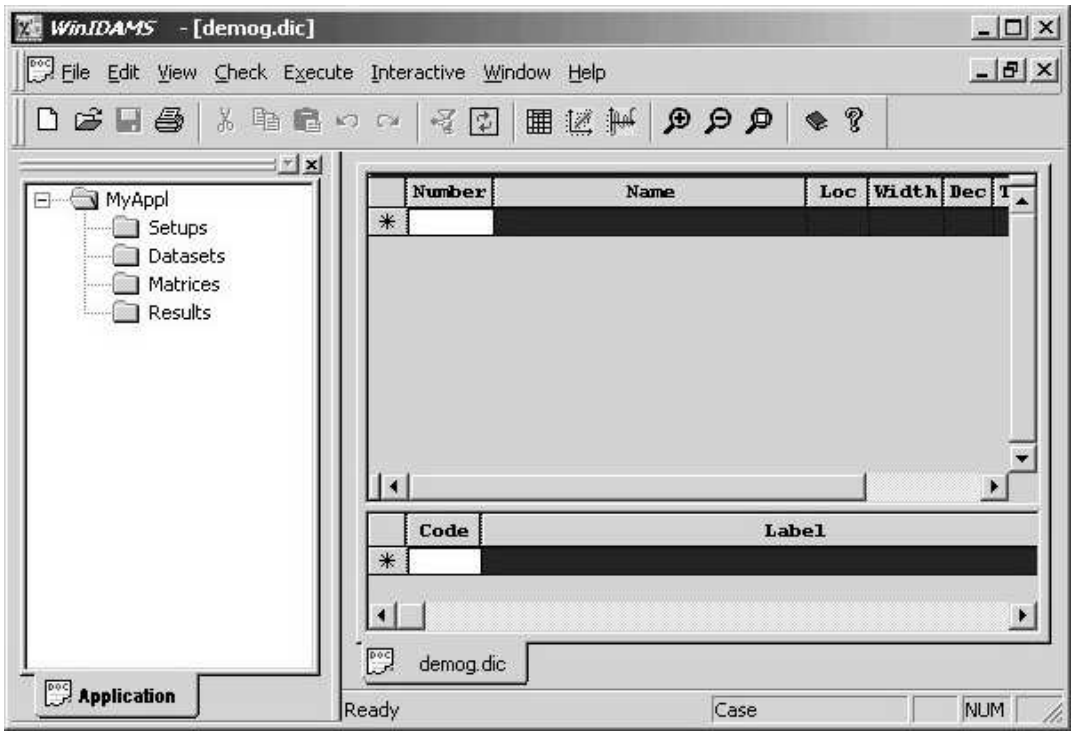
Examples:

```
Data folder:                    c:\MyStudy\students\data
Specification in the setup:     dictin=students2004.dic
Complete dictionary file name:  c:\MyStudy\students\data\students2004.dic
```

9.4 Creating/Updating/Displaying Dictionary Files

The Dictionary window to create, update or display an IDAMS dictionary is called when:

- you create a new Dictionary file (the menu command File/New/IDAMS Dictionary file or the toolbar button New),
- you open a Dictionary file (with extension .dic) displayed in the Application window (double-click on the required file name in the "Datasets" list),
- you open a Dictionary file (with any extension) which is not in the Application window (the menu command File/Open/Dictionary or the toolbar button Open).



This window provides two panes: one for the variable definitions (Variables pane) and another for the codes and code labels of the current variable (Codes pane). A blue line at the top of each pane indicates which pane is active.

The column headings in the Variables pane have following meaning:

Number	Variable number.
Name	Variable name.
Loc, Width	Starting location and field width of the variable in the Data file.
Dec	Number of decimal places; blank implies no decimal places.
Type	Type of variable (N = numeric, A = alphabetic).
Md1	First missing data code for numeric variables.
Md2	Second missing data code for numeric variables.
Refe	Reference number.
StId	Study ID.

For more details, see section "The IDAMS Dictionary" in "Data in IDAMS" chapter. Note that only dictionaries describing data with one record per case can be created, updated or displayed using the Dictionary window.

Changing the pane appearance. The appearance of each pane can be changed separately and the changes apply exclusively to the active pane.

The following modification possibilities are available in each pane:

- *Increasing the font size* - use the toolbar button Zoom In.
- *Decreasing the font size* - use the toolbar button Zoom Out.
- *Resetting default font size* - use the toolbar button 100%.
- *Increasing/Decreasing the width of a column* - place the mouse cursor on the line which separates two columns in the column heading until the cursor becomes a vertical bar with two arrows and move it to the right/left holding the left mouse button.

The Variables pane can further be modified as follows:

- *Increasing/Decreasing the height of rows* - place the mouse cursor on the line which separates two rows in the row heading until the cursor becomes a horizontal bar with two arrows and move it down/up holding the left mouse button.

Defining a variable. Place the cursor in the Variables pane, fill the variable number (at least one is mandatory, subsequent variables will be numbered by adding the value 1), name (optional), location (if not supplied, 1 will be assigned to the first variable and for subsequent variables, location will be calculated by adding the width of the preceding variable) and width (mandatory). Other fields have default values (which you can either accept or modify) or they are optional and can be left blank. Press Enter or Tab to accept a value in a field and move to the next field, or Shift/Tab to move to the previous field. Note that as long as a little pencil appears in the row heading, the row is not saved. Press Enter to accept the complete variable definition. An asterisk in the row heading indicates that this is the next row and you can enter a new variable description.

Defining the codes and code labels for a variable. Switch to the Codes pane and fill the code and label fields. Fill in the code value, then press Enter or Tab and fill the code label, then Enter or Tab to accept the row and move to the next row. When all codes and labels have been defined, switch back to the Variables pane to continue with another variable definition.

Modifying a field in either Variables pane or in Codes pane. Click the field and enter the new value (entering the first character of the new value clears the field). After a double-click on a field, its current value can be partly modified. The Esc key may be used to recuperate previous value.

Editing operations can be performed on one row or on a block of rows. To mark one row, click any field of this row. A triangle appears in the row heading and the row is coloured in dark blue. To mark a block of rows, place the mouse cursor in the row heading where you want to start marking and click the left mouse button. The row becomes yellow, indicating that it is active. Then move the mouse cursor up or down to the row where you want to end marking and click the left mouse button holding the Shift key. Marked rows become dark blue, and the yellow colour shows the active row.

You can Cut, Copy and Paste marked row(s) using the Edit commands, equivalent toolbar buttons or shortcut keys Ctrl/X, Ctrl/C and Ctrl/V respectively.

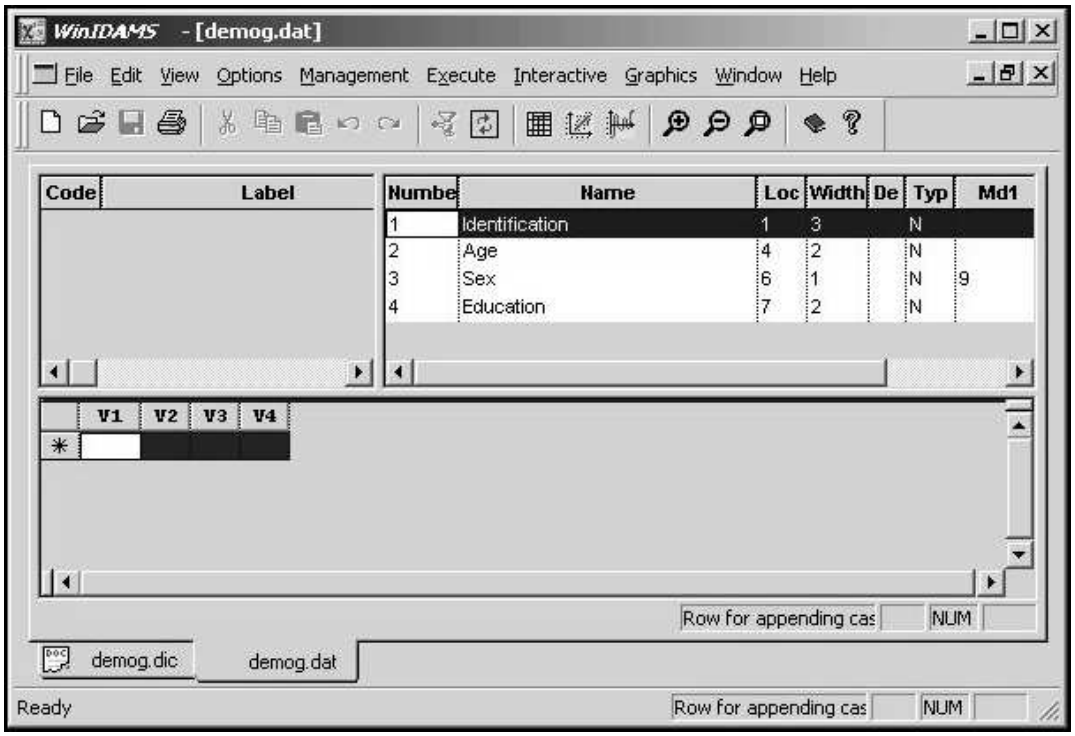
Using the right mouse button you can Insert Before, Insert After, Delete or Clear the active row (even when a block of rows is marked).

Detecting errors in a dictionary. Use the menu command Check/Validity. Errors are signaled one by one and can be corrected once they have all been displayed. Moreover, Interface tries to prevent you from saving dictionaries with errors. Also, when you open a dictionary with errors, their presence is signaled before the dictionary is actually opened.

9.5 Creating/Updating/Displaying Data Files

The Data window is used to create, update or display an IDAMS Data file. Note that the corresponding Dictionary file must already have been constructed and that only Data files with one record per case can be created, updated or displayed using the Data window. This window is called when:

- you create a new Data file (the menu command File/New/IDAMS Data file or the toolbar button New),
- you open a Data file (with extension .dat) displayed in the Application window (double-click on the required file name in the "Datasets" list),
- you open a Data file (with any extension) which is not in the Application window (the menu command File/Open/Data or the toolbar button Open).


WinIDAMS - [demog.dat]
File Edit View Options Management Execute Interactive Graphics Window Help
Code Label
Numbe Name Loc Width De Typ Md1
1 Identification 1 3 N
2 Age 4 2 N
3 Sex 6 1 N 9
4 Education 7 2 N
V1 V2 V3 V4
Row for appending cas NUM
demog.dic demog.dat
Ready
Row for appending cas NUM

The window is divided into 3 panes: one displaying the codes and code labels of the current variable (Codes pane), the second displaying variable definitions (Variables pane) and the third providing place for data entry/modification (Data pane). Only the Data pane can be edited. The other two panes just display the relevant information. A blue line at the top of each pane indicates which pane is active. The panes are synchronized, i.e. selection of a variable field in the Data pane highlights the corresponding variable description, and selection of a field in the Variables pane shows the corresponding variable value in the current case. For the selected variable, codes and code labels (if any) are always displayed.

Changing the pane appearance. The appearance of each pane can be changed separately and the changes apply exclusively to the active pane.

The following modification possibilities are available in all panes:

- *Increasing the font size* - use the menu command View/Zoom In or the toolbar button Zoom In.
- *Decreasing the font size* - use the menu command View/Zoom Out or the toolbar button Zoom Out.
- *Resetting default font size* - use the menu command View/100% or the toolbar button 100%.
- *Increasing/Decreasing the width of a column* - place the mouse cursor on the line which separates two columns in the column heading until the cursor becomes a vertical bar with two arrows and move it to the right/left holding the left mouse button.

The Data pane can be modified further as follows:

- *Increasing/Decreasing the height of rows* - place the mouse cursor on the line which separates two rows in the row heading until the cursor becomes a horizontal bar with two arrows and move it down/up holding the left mouse button.

- *Placing column(s) at the beginning* - mark the required column(s) and use the menu command View/Freeze Columns (use the menu command View/Unfreeze Columns to put them back).
- *Displaying data in a multiple pane* - use the menu command Window/Split. You are provided with a cross to determine the size of four panes. This size can be changed later using the standard Windows technique. Your entire data are displayed four times. The horizontal split can be removed by a double-click on the horizontal line, the vertical split can be removed by a double-click on the vertical line, and the whole split can be removed by a double-click on the split centre.

Entering a new case. Click the first field in an empty row and start entering data values. Press Enter or Tab to accept a data value for the variable and move to the next variable, or Shift/Tab to move to the previous variable. Note that as long as a little pencil appears in the row heading, the case is not saved. Pressing Enter on the last variable saves the case and moves the cursor to the beginning of next row. A new row can be inserted before or after the highlighted row (click on the right mouse button), or can be added at the end of file (row with an asterisk in the row heading).

Data entry can be facilitated taking advantage of two options given in the Options manu:

Code Checking checks data values during data entry against codes defined in the dictionary, being the only codes considered valid.

AutoSkip moves the cursor automatically to the next field once enough digits have been entered to fill the field. If not selected, you have to press Enter or Tab to move to the next field.

Modifying a variable value. Click the variable field and enter the new value (entering the first character of the new value clears the field). A double-click on a variable field can be used to modify part of the current value. The Esc key may be used to recuperate the previous value.

Copying a variable value to another field. Click the variable field and copy its content to the Clipboard (Edit/Copy command, Ctrl/C or Copy button in the toolbar). Then click the required field and paste the value (Edit/Paste command, Ctrl/V or Paste button in the toolbar). The menu command Edit/Undo Case may be used to recuperate the previous value.

Editing operations on one row or on a block of rows can be performed in the same way as in the Dictionary window. To mark one row, click any field of this row. A triangle appears in the row heading and the row is coloured in dark blue. To mark a block of rows, place the mouse cursor in the row heading where you want to start marking and click the left mouse button on. The row becomes yellow, indicating that it is active. Then move the mouse cursor up or down to the row where you want to end marking and click the left mouse button holding the Shift key. Marked rows become dark blue, and the yellow colour shows the active row.

You can Cut, Copy and Paste marked row(s) using the Edit commands, equivalent toolbar buttons or shortcut keys Ctrl/X, Ctrl/C and Ctrl/V respectively.

Using the right mouse button you can Insert Before, Insert After, Delete or Clear the active row (even when a block of rows is marked).

Two data management commands are provided in the Management menu to allow for data verification and sorting:

Check Codes checks data values for all cases in the Data file against codes defined in the dictionary, being the only codes considered valid. At the end of verification, a message showing the number of errors found is displayed and you are invited to correct them one by one using the data correction dialogue box. This box provides case sequential number, variable number and name, invalid code value and a drop-down list of valid codes as defined in the dictionary.

Sort calls the sort dialogue box to specify up to 3 sort variables and corresponding sort order for each of them. After clicking OK, the sorted file appears in the Data pane.

Sorting the data on **one** variable (one column) can also be done by a double-click on the variable number in the Data pane heading. One double-click sorts cases in ascending order. To get the sort in descending order, repeat the double-click.

Two types of graphics are proposed for a variable in the menu Graphics.

Bar Chart provides a bar chart based on either frequencies or percentages for qualitative variable categories. For quantitative variables, the user defines the number of bars (NB) on both sides of the mean (M) and a coefficient (C) for calculating bar (class) width. The bar width (BW) is equal to the value of standard deviation (STD) multiplied by the coefficient (BW=C*STD). The bars are constructed using the values M-NB*BW, ..., M-2BW, M-BW, M, M+BW, M+2BW, ..., M+NB*BW. The height of a rectangle = (relative frequency of class)/(class width). In addition, normal distribution curve having the calculated mean and standard deviation can be projected for quantitative variables.

Histogram, meant for quantitative variables, provides a histogram based either on frequencies or on percentages with the number of bins specified by the user.

Graphics for quantitative variables contain also univariate statistics for the projected variable such as: mean, standard deviation, variance, skewness and kurtosis. Variables with decimal places are multiplied by a scale factor in order to obtain integer values. In this case, mean value, standard deviation and variance should be adjusted accordingly.

9.6 Importing Data Files

WinIDAMS provides a tool for importing data files to IDAMS directly through the WinIDAMS User Interface. This facility can be accessed in the WinIDAMS Main window, the Data window and the Multidimensional Tables window.

Three types of free format files can be imported:

- .txt files in which fields are separated by tabs,
- .csv files in which fields are separated by commas,
- .csv files in which fields are separated by semicolons.

Information provided in the first row is considered to be column labels and is used as variable names during the dictionary construction process. Thus, the presence of column labels is mandatory in the first row of input files.

Also the separation character is determined from the first line while the character used as decimal separator is detected from the second line (first data line) of the file. Thus, if a variable is expected to have decimal values, it should be shown in the first data line.

During the import process, contents of imported alphabetic variables can be changed to numeric codes, keeping the alphabetic values as code labels in the created IDAMS dictionary. Commas used as decimal separator for numeric variables are changed to points.

The Data Import operation is activated with the command File/Import, followed by selection of required file in the standard file Open dialogue box. The separation character and the character used as decimal separator are displayed together with values of all fields for the first three cases. Data reading can then be checked before launching the import. Afterwards, you are provided with two windows called External data and Variables Definition, both having form of a spreadsheet.

The External data window only displays the contents of the file to import. No editing operations are allowed, except copying a selection to the Clipboard.

The Variables Definition window serves for preparing IDAMS variable descriptions. Its initial content is provided by default and on the basis of the imported data, but you are free to change and to complete it as necessary.

The columns contain the following information:

Description	Variable name.
Type	Type of variable (numeric by default). This is the input variable type. If an input variable is alphabetic and should be output as numeric, ask for recoding (see below).

MaxWidth	Maximum field width of the variable.
NumDec	Number of decimal places; blank implies no decimal places.
Md1	First missing data code for numeric variables.
Md2	Second missing data code for numeric variables.
Recoding	Requesting a recoding of alphabetic variables to numeric values.

To modify variable definitions, place the cursor inside the window. Then use the navigation keys or the mouse to move to the required field and change its contents.

Use the menu command Build/IDAMS Dataset to create IDAMS Dictionary and Data files. They will both be placed in the Data folder of the current application.

9.7 Exporting IDAMS Data Files

WinIDAMS also has a tool for exporting IDAMS Data files directly through the WinIDAMS User Interface. This can be done from the Data window using the command File/Export. The IDAMS Data file displayed in the active window can be saved in one of the three types of free format data files:

- .txt files in which fields are separated by tabs,
- .csv files in which fields are separated by commas,
- .csv files in which fields are separated by semicolons.

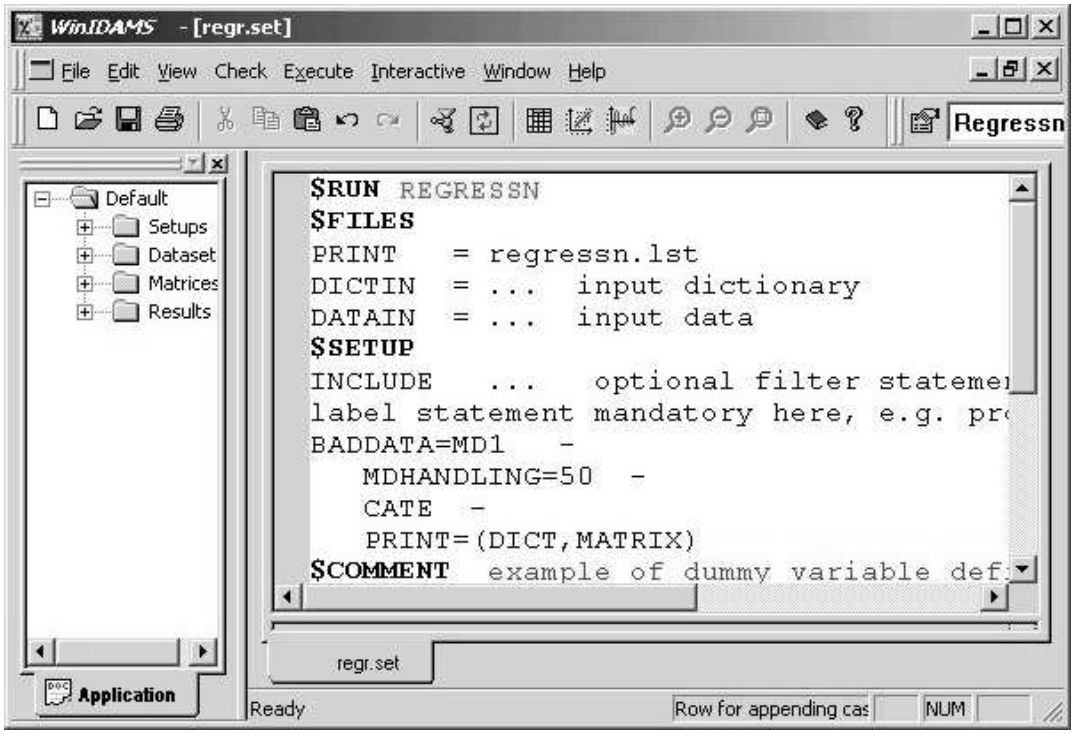
Variable names from the corresponding Dictionary file are output in the first row of the exported data as column labels.

If code labels exist for a variable, numeric code values can be optionally replaced by their corresponding code label in the output data file. Moreover, numeric variables can be output with comma used as decimal separator.

9.8 Creating/Updating/Displaying Setup Files

The Setup window to prepare or to display an IDAMS Setup file is called when:

- you create a new Setup file (the menu command File/New/IDAMS Setup file or the toolbar button New),
- you open a Setup file (with extension .set) displayed in the Application window (double-click on the required file name in the “Setups” list),
- you open a Setup file (with any extension) which is not in the Application window (the menu command File/Open/Setup or the toolbar button Open).



The window provides two panes: the top one is for preparing the Setup file itself (Setup pane) and the bottom one for displaying error messages when filter and Recode statements are checked (Messages pane). Only the Setup pane can be edited. Note that IDAMS commands are displayed in bold and program names in pink if they are spelled correctly. Text put on a $comment command is displayed in green.

To prepare a new program setup, you can either type in all statements or you can use the prototype setup for the required program and modify it as necessary. Prototype setups are provided for all programs. They can be accessed by selecting the program name in the list under the toolbar button Prototype. To copy the prototype to the Setup pane, click the required program name. For details on how to prepare setups, see the chapter "The IDAMS Setup File" and the relevant program write-up.

Editing operations can be performed as with any ASCII file editor, i.e. you can Cut, Copy and Paste any selection, using the Edit commands, equivalent toolbar buttons or shortcut keys Ctrl/X, Ctrl/C and Ctrl/V respectively.

Two setup verification commands are provided in the Check menu to allow for syntax verification of sets of Recode statements and filter statements:

Recode Syntax activates verification of syntax in Recode statements included in the setup. All errors found are reported in the Messages pane giving the Recode set number, erroneous statement line and character(s) causing the syntax problem. A double-click on the erroneous line text or on the error message in the Message pane shows this line in the Setup pane with a yellow arrow. You can correct the errors and repeat syntax verification, before passing the setup for execution.

Filter Syntax activates verification of syntax errors in filter statements included in the setup. All errors found are reported in the Messages pane giving the filter statement number, erroneous statement line and character(s) causing the syntax problem. A double-click on the erroneous line text or on the error message in the Messages pane shows this line in the Setup pane with a yellow arrow.

Note that although most syntax errors in filter and Recode statements can be detected and corrected here, another syntax verification is systematically performed by IDAMS during setup execution. Also execution errors, which cannot be detected here, are reported in the results.

9.9 Executing IDAMS Setups

To execute IDAMS program(s) (for which instructions have been prepared and saved in a Setup file), use the menu command Execute/Select Setup in any WinIDAMS document window. You are asked, through the standard Windows dialogue box, to select the file from which instructions should be taken for execution.

If you are preparing your instructions in the Setup window, you can execute programs from the Current Setup using the menu command Execute/Current Setup.

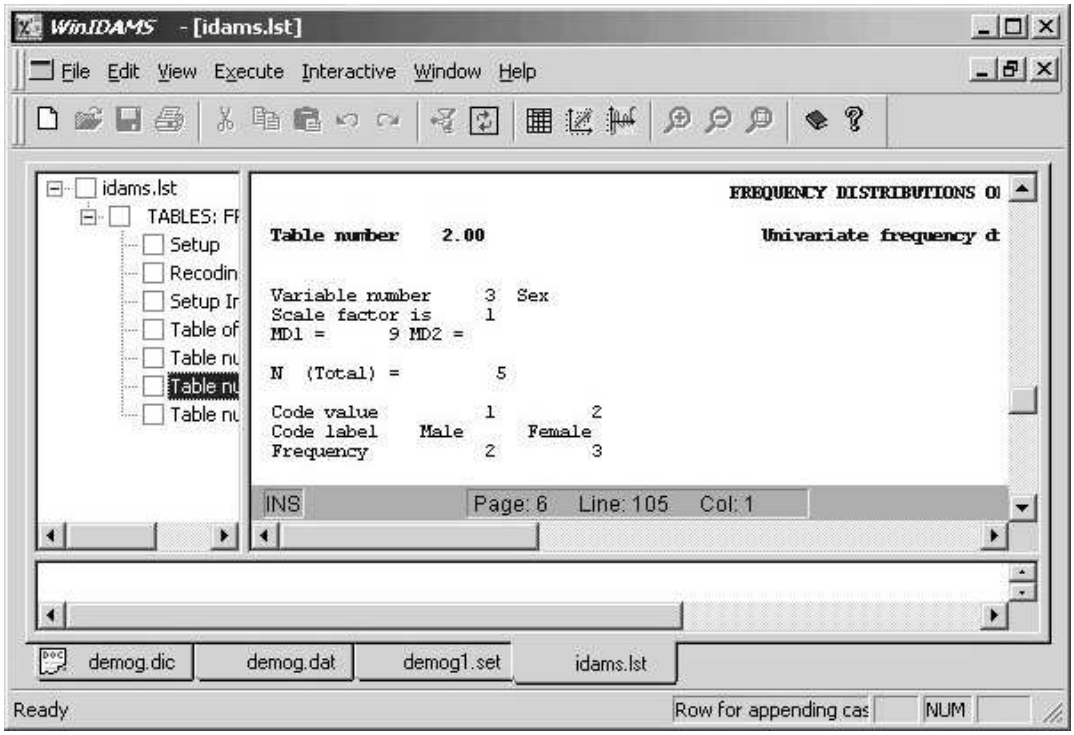
The program(s) will be executed and the results written to the file specified for PRINT under $FILES (the default is IDAMS.LST in the current Work folder). At the end of execution, the Results file will be opened in the Results window.

9.10 Handling Results Files

The Results window to access, display and print selected parts of the results is called when:

- you open a Results file (with extension .lst) displayed in the Application window (double-click on the required file name in the "Results" list),
- you open a Results file (with any extension) which is not in the Application window (the menu command File/Open/Results or the toolbar button Open),
- you execute IDAMS setup; the contents of the Results file is displayed automatically.

Quick navigation in the results is facilitated through their table of contents. You can access the beginning of particular program results or even a particular section. Moreover, the menu Edit provides access to a searching facility.



The window is divided into 3 panes: one showing the table of contents (TOC) of the results as a structure tree, the second displaying the results themselves and the third displaying error messages and warnings included in the results.

By default, the pagination of results done by programs is retained (the **Page Mode** option in the check box of **View** menu is marked). To make the results more compact, unmark this option. Trailing blank lines will be removed from all pages and page breaks inserted by programs will be replaced by "Page break" text line.

To open/close quickly the TOC tree, three buttons on the numeric pad are available:

*	opens all levels of the tree under the selected node
-	closes all levels of the tree under the selected node
+	opens one level under the selected node.

To view a particular part of the results, double-click on its name in the TOC.

To locate an error message or a warning, double-click its text.

Modification of the results is not allowed. However, selected parts (highlighted or marked in tick-boxes in the TOC tree) or all the results can be copied to the Clipboard (Edit/Copy command, Ctrl/C or Copy button in the toolbar) and pasted to any document using standard Windows techniques.

Printing the whole contents or selected pages of the results can be done through the menu command File/Print or using the Print toolbar button. Note that printing is done in Landscape orientation, and this orientation cannot be changed.

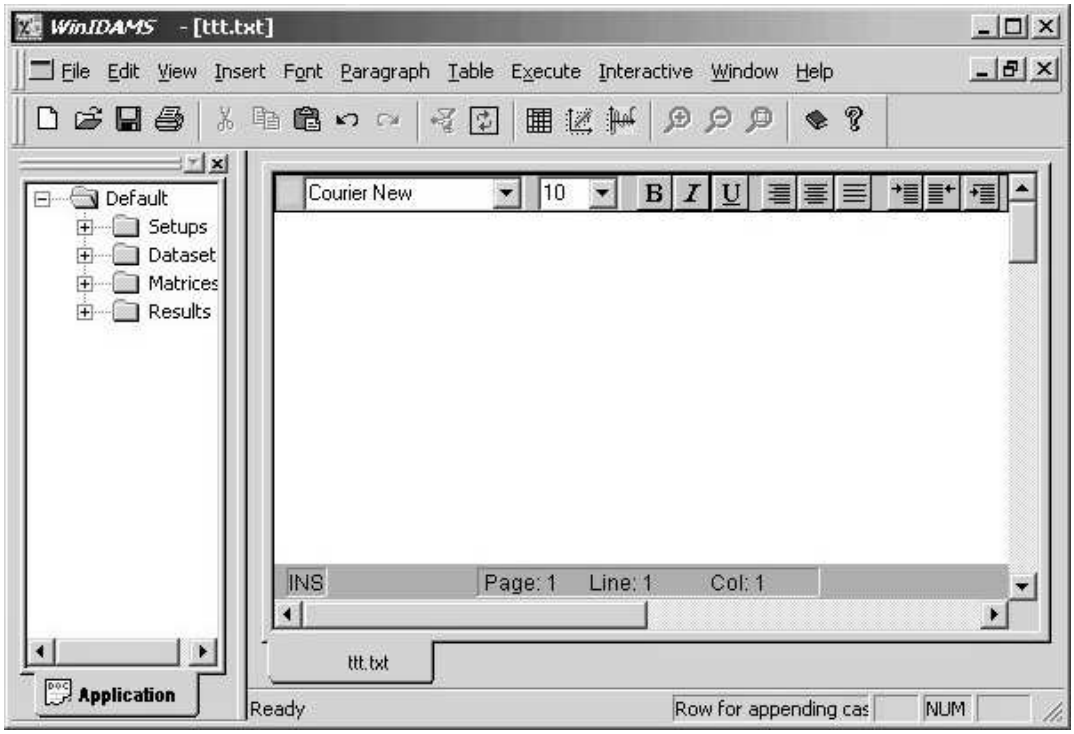
The contents of the Results file as displayed can be **saved** in RTF or in text format using the menu command File/Save As. Trailing blank lines are always removed. Page breaks are handled according to the Page Mode option.

9.11 Creating/Updating Text and RTF Format Files

WinIDAMS has a General Editor which allows you to open and modify any type of document in character format. However, its basic function is to provide a facility for editing Text files and to offer sophisticated formatting and editing features. Manipulation of Dictionary, Data or Setup files using the General Editor should be avoided, and manipulation of Matrix files should be performed with caution.

The Text window is called when:

- you create a new Text file (the menu command File/New/Text file or RTF file, or the toolbar button New),
- you open a Matrix file (with extension .mat) displayed in the Application window (double-click on the required file name in the "Matrices" list),
- you open any character file which is not in the Application window (the menu command File/Open/File Using General Editor or the toolbar button Open).



The General Editor provides a number of standard editing commands which are known to Windows users. They are listed below but will not be described in detail.

Insert provides commands for inserting page and section breaks, picture, OLE object (Object Linking & Embedding), frame and drawing object.

Font commands allow you to change font and colour of selected text, and the colour of its background.

Paragraph commands enable you user to align paragraphs differently, to indent them, to display them in double space, and to draw a border around and shade the background.

Table gives access to a number of commands to insert and manipulate tables.

View contains three additional commands to display the active document in page mode, to display the ruler and the paragraph marker.

Formatting toolbar allows you to choose quickly formatting commands that are used most frequently.

Part III

Data Management Facilities

Chapter 10

Aggregating Data (AGGREG)

10.1 General Description

AGGREG aggregates individual records (data cases) into groups defined by the user and computes summary descriptive statistics on specified variables for each group. The statistics include sums, means, variances, standard deviations, as well as minimum and maximum values and the counts of non-missing data values. An output IDAMS dataset is created, i.e. the grouped (aggregated) data file described by an IDAMS dictionary; the aggregated data file contains one record (case) per group with variables that are the summary to the group level of each of the selected input variables.

Formulas for calculating mean, variance and standard deviation can be found in Part "Statistical Formulas and Bibliographic References", chapter "Univariate and Bivariate Tables". However, they need to be adjusted since cases are not weighted and the coefficient N/(N-1) is not used in computation of sample variance and/or standard deviation. Note that the summary statistics are selected for the entire set of aggregate variables. Thus, if there were 2 aggregate variables and if 3 statistics were selected, there would be 6 computed variables.

AGGREG enables the user to change the level of aggregation of data e.g. from individual family members to household, or from district to regional level, etc. For example, suppose a data file contains records on every individual in a household and that we wish to analyze these data at the household level. AGGREG would permit us to aggregate values of variables across all the individual records for each household to create a file of household level records for further analysis. If, to be more specific, the individual level data file contained a variable giving the persons income, AGGREG could create household level records with a variable on the total household income.

Grouping the data. The user specifies up to 20 group definition (ID) variables which determine the level of aggregation for the output file. For example, if one wanted to aggregate individual level data to the household level, a variable identifying the household would be the group definition variable. Each time AGGREG reads an input record, it checks for a change in any of the ID variables. When this is encountered, a record is output containing the summary statistics on the specified aggregate variables for the group of records just processed.

Inserting constants into the group records. Constants can be inserted into each group record using the parameters PAD1, ... , PAD5, which specify so called pad variables. The value of a pad variable is a constant.

Transferring variables. Variables can be transferred to the output group records. Note that only the values of the first case in the group are transferred.

10.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of the cases from the input data. ID variables defining the groups and the variables to be aggregated are specified with the parameters. The ID variables are automatically included in the output group dataset.

Transforming data. Recode statements may be used.

Treatment of missing data. Each aggregate variable value is compared to both missing data codes and if found to be a missing data value, is automatically excluded from any calculation. A user-supplied percentage, the "cutoff point" (see the parameter CUTOFF) determines the number of missing data values allowed before the summarization value is output as a missing data code. Thus, for example, suppose the mean value of an aggregate variable within a group was to be computed, and the group contained 12 records and 6 of them had missing data values, i.e. 50%. If the CUTOFF value was 75%, the mean of the 6 non-missing values would be calculated and output for that group. If the CUTOFF value was 25%, however, the mean would not be calculated and the first missing data code would be output.

10.3 Results

Missing data summary. (Optional: see the parameter PRINT). For each variable in each group, the input variable number, the output variable number, the number of records with substantive data (i.e. non-missing data) and the percentage of records with missing data are printed.

Group summary. (Optional: see the parameter PRINT). The number of input records for each group.

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Output dictionary. (Optional: see the parameter PRINT).

Statistics. (Optional: see the parameter PRINT). All of the computed variables can be printed for each aggregate record. The variable number of the corresponding aggregate variable and the ID variables are also given.

10.4 Output Dataset

The grouped output dataset is a Data file, described by an IDAMS dictionary. Each record contains values of the ID variables, computed variables, transferred variables and pad constants; there is one record produced for each group.

Variable sequence and variable numbers. The output variables are in the same relative order as the input variables from which they were derived, regardless of whether the input variable is used as an ID, aggregate, or variable to be transferred. Thus, if the first variable in the input is used, the variable(s) derived from it will be the first output variable(s). Each input variable used as an ID or variable to be transferred corresponds to one output variable; each aggregate variable corresponds to from 1 to 7 output variables, according to the number of summary statistics requested (these variables are output in the relative order: sum, mean, variance, standard deviation, count, minimum, maximum). The output variables are always renumbered, starting with the number supplied in the parameter VSTART. Pad constants always come last.

Variable names. The output variables have the same names as input variables from which they were derived except that for the aggregate variables, the 23rd and 24th characters of the name field are coded:

S	=	sum
M	=	mean
V	=	variance
D	=	standard deviation
CT	=	count
MN	=	minimum
MX	=	maximum.

Pad constants are given names "Pad variable 1", "Pad variable 2", etc.

Variable type. ID variables and transferred variables are output in their input type. Computed variables are always output as numeric.

Field width and number of decimals. Field widths for output aggregated variables depend on the statistic, the input field width (FW), the input number of decimal places (ND) and the extra decimal places

requested by the user with the DEC parameter. Field widths and decimal places are assigned as shown below, where FW=input field width and ND=input number of decimal places for input variables, and FW=6 and ND=0 for recoded variables.

Statistic	*Field Width*	*Decimal Places*
SUM	FW + 3 *	ND
MEAN	FW + DEC **	ND + DEC ***
VARIANCE	FW + DEC **	ND + DEC ***
SD	FW + DEC **	ND + DEC ***
MIN	FW	ND
MAX	FW	ND
COUNT	4	0

*	If the field width exceeds 9, then it is reduced to 9.
**	If the field width exceeds 9, then the number of extra decimals (DEC) is reduced accordingly.
***	If the number of decimals exceeds 9, then DEC is reduced accordingly.

Missing data codes. Missing data codes for ID variables and transferred variables are taken from the input dictionary. The second missing data code (MD2) for the computed variables is always blank. The value of the first missing data code (MD1) is allocated as follows:

Output variable	*Output MD1*
Output FW <= 7	9's
Output FW > 7	-999999
COUNT variable	9999

Reference numbers. Computed variables are given the reference number of their base variable.

C-records. C-records in the input dictionary are transferred to the output dictionary for ID and transfer variables.

A note on computation of the statistics. Before output, computed values are rounded up to the calculated width and number of decimal places. If the computed value exceeds 999999999 or is less than -99999999, it is output as 999999999.

10.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. Group-definition (ID) variables and variables to be transferred may be numeric or alphabetic, although numeric variables are treated as strings of characters, i.e. a value of '044' is different from ' 44'. They cannot be recoded variables. Variables to be aggregated must be numeric and may be recoded variables.

The file is processed serially and contiguous records with the same value on the ID variables are aggregated. Thus, the input file should be sorted on the ID variables prior to using AGGREG. Note that AGGREG does not check the input file sort order.

10.6 Setup Structure

```
$RUN AGGREG

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default IDAMS.LST)
```

10.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V1=10,20,30,50 OR V10=90-300
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  AGGREGATION TEACHER/STUDENT DATA
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  IDVARS=(V1,V2)  STATS=(SUM,VARI)  DEC=3 -
                AGGV=(V5-V10,V50-V75)  PAD1=80
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   BADDATA=**STOP**/SKIP/MD1/MD2
   Treatment of non-numeric data values in aggregates variables and in variables used in Recode. See "The IDAMS Setup File" chapter.

MAXCASES=n
: The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

IDVARS=(variable list)
: Up to 20 variable numbers to define the groups. R-variables are not allowed.
No default.

AGGV=(variable list)
: V- or R-variables to be aggregated.
No default.

STATS=(SUM, MEAN, VARIANCE, SD, COUNT, MIN, MAX)
: Parameters for selecting required statistics (at least one of: SUM, MEAN, VARIANCE, SD must be selected). They are output for each group and for each AGGV variable.

SUM	Sum.
MEAN	Mean.
VARI	Variance.
SD	Standard deviation.
COUN	Number of valid cases.
MIN	Minimum value.
MAX	Maximum value.

SAMPLE/POPULATION

SAMP	Compute the variance and/or standard deviation using the sample equation.
POPU	Use the population equation.

OUTFILE=**OUT**/yyyy
: A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

VSTART=**1**/n
: Variable number for the first variable in the output dataset.

CUTOFF=**100**/n
: The percentage of cases with MD codes allowed before a MD code is output. An integer value.

DEC=**2**/n
: For computed variables involving mean, variance or standard deviation: the number of decimal places in addition to those of the corresponding input variables (see Restriction 7).

TRANSVARS=(variable list)
: Variables whose values, as given for the **first** case of each group, are to be transferred to the output file. R-variables are not allowed.

PAD1=constant
PAD2=constant
PAD3=constant
PAD4=constant
PAD5=constant
: Up to 5 constants can be added to the output dataset. The number of characters given determines the field width of the constant.

PRINT=(MDTABLES, GROUPS, DATA, CDICT/DICT, **OUTDICT**/OUTCDICT/NOOUTDICT)

MDTA	Print a table giving the percentage of missing data found for each aggregate variable in each group.
GROU	Print the number of cases per group.
DATA	Print values for each computed variable in each group record.
CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records of ID and transfer variables if any.
NOOU	Do not print the output dictionary.

10.8 Restrictions

1. Maximum number of variables to be aggregated is 400.
2. Maximum number of ID variables is 20.
3. Maximum number of characters in ID variables is 180.
4. Maximum number of variables to be transferred is 100.
5. Recoded variables are not allowed as IDVARS or as TRANSVARS.
6. Same variable cannot appear in two variable lists.

10.9 Example

Output a dataset containing one aggregate case for each unique value of V5 and V7; the variables in each case are to be the sum, mean and standard deviation of 4 input variables and 1 recoded variable, aggregated over the cases forming the group (i.e. with the same values for V5, V7); values of V10, V11 for the first case of each group are to be transferred to the output records; a listing of the values output for each case is requested; in the output file, variables are to be numbered starting from 1001.

```
$RUN AGGREG
$FILES
PRINT   = AGGR.LST
DICTIN  = IND.DIC                       input Dictionary file
DATAIN  = IND.DAT                       input Data file
DICTOUT = AGGR.DIC                      output Dictionary file
DATAOUT = AGGR.DAT                      output Data file
$RECODE
    R100=COUNT(1,V20-V29)
    NAME R100'WEALTH INDEX'
$SETUP
AGGREGATION OF 4 INPUT VARIABLES AND 1 RECODED VARIABLE
IDVARS=(V5,V7)  AGGV=(V31,V41-V43,R100)  STATS=(SUM, MEAN, SD)   -
   VSTART=1001  PRINT=DATA  TRANS=(V10,V11)
```

Chapter 11

Building an IDAMS Dataset (BUILD)

11.1 General Description

BUILD takes a raw data file, which may contain several records per case, along with a dictionary describing the required variables and creates a new Data file with a single record per case containing values only for the specified variables. At the same time, it outputs an IDAMS dictionary describing the newly formatted Data file, in other words an IDAMS dataset is created.

In addition to restructuring the data, BUILD also checks for non-numeric values in numeric variables.

Why use BUILD? Any IDAMS program can be used without first using BUILD by preparing separately an IDAMS dictionary. However BUILD is recommended as a preliminary step since it:

- provides checks on the correct preparation of the dictionary,
- ensures that there is an exact match between the dictionary and the data,
- ensures that there are no unexpected non-numeric characters in the data,
- reduces the data into a compact single record per case form,
- recodes all blank fields to user specified values.

Numeric variable processing. When BUILD processes a field as containing a numeric variable, it checks that the field either contains a recognizable number or is blank. If a value other than these occurs, e.g. '3J', '3-', '**2', etc. the sequential position of the case, the variable number associated with the field, and the input case are printed and a string of nines is used as the output value.

Processing rules are as follows:

- If a field contains a recognizable number, the number is edited into a standard form and output (see the "Data in IDAMS" chapter for details).
- If a field contains all blanks, it is either recoded to the 1st or 2nd missing data code, nines or zeros, or, if no recoding is specified, it is signaled as an error and output as blank field. Column 64 of T-records may be used to specify recoding rule for the variable (see "Input Dictionary" section for details).
- If a field contains illegal trailing blanks, e.g. '04 ' in a three digit numeric field, or embedded blanks, e.g. '0 4', it is reported as error and the value is changed to 9's.
- If a field contains a positive value or a negative value with the '+' or '-' characters wrongly entered, e.g. '1-23', it is reported as error and the value is changed to 9's.
- If a missing data code for a variable has one more digit than the input field, the output field will be one character longer than the input. This feature can be used when it is necessary to increase the output field width without changing the input field width; for example, if codes 0-9 and a blank were defined for a single column variable, the blank field could not be recoded to a unique numeric value without allowing a 2-digit code on output.

Table showing examples of editing performed by BUILD
and the contents of the output field for a 3-digit input numeric field

Input value	No. dec.	MD1	Recoding specified	Output value	Output field width	Error message
032	0	9999	-	0032	4	-
32	0		-	032	3	-
3 2	0		-	999	3	embedded blanks in var ...
32	0		-	999	3	embedded blanks in var ...
-03	0		-	-03	3	-
-3	0		-	-03	3	-
- 3	0		-	-03	3	-
3.2	0		-	003	3	-
32	1		-	032	3	-
.32	1		-	003	3	-
3.2	1		-	032	3	-
.32	2		-	032	3	-
.35	1		-	004	3	-
-.3	0		-	-00	3	-
-.3	1		-	-03	3	-
-03	1		-	-03	3	-
	-	8888	1	8888	4	(only if PRINT=RECODES)
	-		0	000	3	(only if PRINT=RECODES)
	-		None		3	blanks in var ...
A32	-		-	999	3	bad characters in var ...
3-2	-		-	999	3	bad characters in var ...

11.2 Standard IDAMS Features

Case and variable selection. This program has no provision for selecting cases from the input data file. The standard filter is not available. By way of the variable descriptions, any subset of the fields within a case may be selected for the output data.

Transforming data. Recode statements may not be used.

Treatment of missing data. BUILD makes no distinction between substantive data and missing data values. However, blank fields may be replaced by missing data codes, zeros or nines.

11.3 Results

Input dictionary. (Optional: see the parameter PRINT). "Brule" column on the dictionary listing contains recoding rules for blank fields, as specified in col. 64 of the input dictionary. Note that error messages for the dictionary are interspersed with the dictionary listing and do not contain a variable number. If the input dictionary is not printed, the errors may be difficult to identify.

Output dictionary. (Optional: see the parameter PRINT). Variable description records (T-records) are printed without or with C-records, if any.

Output data file characteristic. Record length of the output data file.

Data editing messages. For each case containing errors, the input case (up to 100 characters per line) and a report of errors in variable number order are printed.

Blank field recoding messages. (Optional: see the parameter PRINT). For each case containing blank fields that were recoded, a message about this along with the input data case are printed. These messages are integrated with the data editing messages, if any errors also occur in the case.

11.4 Output Dataset

BUILD creates a Data file and a corresponding IDAMS dictionary, i.e. an IDAMS dataset. Note that the T-records always define the locations of variables in terms of starting position and field width.

The data file contains one record for each case. The record length is the sum of the field widths of all variables output and is determined by the BUILD program.

Numeric variable values. Numeric variable values are edited to a standard form as described in the "Numeric variable processing" paragraph above.

Alphabetic variable values. The data values for alphabetic variables are not edited and are the same on input and output.

Variable width. Normally BUILD assigns the width of a variable to be the same as the number of characters the variable occupies in the input data. However, if a missing data code has one more significant digit than the input field width, the output field width will be increased by one.

Variable location. BUILD assigns the output fields in variable number order. Thus, if the first two variables have output widths of 5 and 3, locations 1-5 are assigned to the first variable and 6-8 are assigned to the second, etc.

Reference number and study ID. The reference number, if it is not blank, and study ID are the same as their input values. If the reference number field of an input T-record or C-record is blank, it is filled with the variable number.

11.5 Input Dictionary

This describes those variables that are to be selected for output. The format is as described in the "Data in IDAMS" chapter with column 64 of T-records being used to specify a recoding rule for blanks in a variable as follows:

blank	-	no recoding of blank fields,
0	-	recode blank fields to zeros,
1	-	recode blank fields to 1st missing data code for variable,
2	-	recode blank fields to 2nd missing data code for variable,
9	-	recode blank fields to 9's.

Note: The Dictionary window of the User Interface does not provide access to the column 64. Thus, use the WinIDAMS General Editor (File/Open/File Using General Editor) or any other text editor to fill in this column.

11.6 Input Data

The data can be any fixed-length record file with one or more records per case providing there are exactly the same number of records for each case. The file should be sorted by record type within case ID. The values for any variable must be located in the same columns in the same record for every case.

If the input data has more than one record per case, MERCHECK should always be used prior to BUILD to ensure that the data do have the same set of records for each case.

Note that the exponential notation of data is not accepted by BUILD.

11.7 Setup Structure

```
$RUN BUILD

$FILES
     File specifications

$SETUP
     1. Label
     2. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default IDAMS.LST)
```

11.8 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-2 below.

1. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  FILE BUILDING STUDY A35
   ```

2. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  MAXERROR=50
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   LRECL=**80**/n
   The length of each input data record.
   (Used to check if variable starting locations on T-records are valid).

   MAXCASES=n
   The maximum number of cases to be used from the input file.
   Default: All cases will be used.

   VNUM=**CONTIGUOUS**/NONCONTIGUOUS
   CONT Check that variables are numbered in ascending order and consecutively in the input dictionary.
   NONC Check only that variables are numbered in ascending order.

MAXERR=**10**/n
: The maximum number of cases with errors (unrecoded blanks and non-numeric values for numeric variables) before BUILD terminates execution.

OUTFILE=**OUT**/yyyy
: A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

PRINT=(RECODES, CDICT/DICT, **OUTDICT**/OUTCDICT/NOOUTDICT)

RECO	Print input cases that contain one or more blank fields which have been recoded.
CDIC	Print the input dictionary for all variables with C-records if any.
DICT	Print the input dictionary without C-records.
OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
NOOU	Do not print the output dictionary.

11.9 Examples

Example 1. Build an IDAMS dataset (dictionary and data file); input data records have a record length of 80 with 3 records per case; variables are numbered non-contiguously in the input dictionary; variable V2 is the complete ID (columns 5-10) while variables V3 and V4 contain the two parts of the ID (columns 5-8, 9-10 respectively); blank fields should be replaced by the first missing data code for variables V101, V122, V168, and by zeros for variable V169; blanks for V123 (age) should be treated as errors.

```
$RUN BUILD
$FILES
DATAIN  = ABCDATA RECL=80              input Data file
DICTOUT = ABC.DIC                      output Dictionary file
DATAOUT = ABC.DAT                      output Data file
$SETUP
BUILDING A IDAMS DATASET
VNUM=NONC  MAXERR=200
$DICT
   3   1 169   3
T   1 TOWN CODE                1 1 1 3                                 ID
T   2 RESPONDENT ID              5  10                                 ID
T   3 HOUSEHOLD NUMBER           5   8                                 ID
T   4 RESPONDENT NUMBER          9  10                                 ID
T 101 RESP POSITION IN FAMILY   13              0      9      1       QS1
T 122 SEX                      225              9             1       QS2
T 123 AGE                       48  49                                 QS2
T 168 OCCUPATION               358  59          99     98      1       QS3
T 169 INCOME                    61  65              99998      0       QS3
```

Example 2. Verify the presence of non-numeric characters in 4 numeric fields; the input data file has one record per case; records are identified by an alphabetic field; the 5 variables are not numbered contiguously; the output files normally produced by BUILD are not required and are defined as temporary files (extension TMP) which are automatically deleted by IDAMS at the end of execution.

```
$RUN BUILD
$FILES
DATAIN  = A:NEWDATA RECL=256            input Data file
DICTOUT = DIC.TMP                       temporary output Dictionary file
DATAOUT = DAT.TMP                       temporary output Data file
$SETUP
CHECKING FOR AND REPORTING NON-NUMERIC CHARACTERS AND BLANKS
VNUM=NONC  LRECL=256  PRINT=NOOU  MAXERR=200
$DICT
   3   1  35   1   1
T   1 RESPONDENT NAME             1  20 1
T  21 AGE                        21   2
T  22 INCOME                     29   6
T  25 NO. WORK PLACES           129   1
T  35 SCI. TITLE                201   1
```

Chapter 12

Checking of Codes (CHECK)

12.1 General Description

CHECK verifies whether variables have valid data values and lists all invalid codes by case ID and variable number.

Code specification. There are two ways in which the codes for the variables to be checked may be specified. First, the program control statements include a set of "code specifications" with which to define the variables and their valid codes. Second, the user may supply a list of variables for which valid codes are to be taken from C-records in the dictionary. In any given execution of CHECK, the user may apply the first method for some variables and the second method for others. Code specifications for a variable in the setup override dictionary specifications.

Method used for checking data values. Data values for variables, both numeric and alphabetic, are checked against the valid codes specified on a character by character basis. Thus, if a valid code specification of 'V2=02,03' is given, then a value of ' 2' in the data will be invalid; a leading blank in the data is not considered equal to a zero. If code values are specified with fewer digits than the field width of the variable, leading zeros are assumed. Thus, if the specification 'V2=2,3' is given where V2 is a 2-digit variable, valid values used for comparison to the data will be taken as 02, 03. Similarly, if '-3' and '1' were supplied as valid codes for a 3-digit variable, CHECK would edit the codes to '-03' and '001' before comparing any data value to them.

Note. If a syntax error is found in a code specification, the other code specifications are checked but the data are not processed.

12.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input dataset. The user selects the variables to be checked either by specifying them on a "variable list" and/or on the "code specifications".

Transforming data. Recode statements may not be used.

Treatment of missing data. CHECK makes no distinction between substantive data and missing data values; all data are treated the same.

12.3 Results

Input dictionary. (Optional: see the parameter PRINT). Dictionary records for all variables are printed, not just for those being checked.

Documentation of invalid codes. For each case in which a variable is found to have an invalid code, CHECK prints the ID variable value(s), the variables in error and their values.

12.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. CHECK can check for valid data on both numeric and alphabetic variables. If the dictionary contains C-records, these can be used to define valid codes for variables.

Values for numeric variables are assumed to be in the form they would have after being edited by BUILD. This assumption implies that there are no leading blanks (they have been replaced by zeros), that a negative sign, if any, appears in the left most position, and that explicit decimal points do not appear.

12.5 Setup Structure

```
$RUN CHECK

$FILES
     File specifications

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Code specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default IDAMS.LST)
```

12.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V10=3 AND V20=1-9
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  DATA: THESIS DATA, VERSION 1
   ```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  IDVA=(V1-V4) VARS=(V22-V26,V101-V102)
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

START=**1**/n
The sequential number of the first case to be checked.

VARS=(variable list)
Variables for which valid codes are to be taken from the C-records in the dictionary.

MAXERR=**100**/n
Maximum number of cases with invalid codes allowed; if this number is exceeded, the execution is terminated.

IDVARS=(variable list)
Up to 20 variables whose value(s) are to be printed when an invalid code is found. These will normally consist at minimum of the variables that identify a case but can include others which will provide additional information to the user. The variables may be alphabetic or numeric.
No default.

PRINT=CDICT/DICT
CDIC Print the input dictionary for all variables with C-records if any.
DICT Print the input dictionary without C-records.

4. **Code specifications** (optional). These specifications define the variables to be checked and their valid or invalid code values.

```
Examples:

V3=1,3,5-9          (The data for variable 3 may have codes 1,3,5-9.
                     Any other code values are invalid and will be documented).

V7,V9,V12-V14= -    (The data for variables 7,9 and 12 through 14
  2,50-75,100        may only have values 2,50-75,100).

V50 <> 75           (The data for variable 50 may have any code except 75).
```

General format

variable list = list of code values
or
variable list <> list of code values

Rules for coding

Each code specification must start on a new line. To continue to another line, break after a comma and enter a dash. As many continuation lines may be used as necessary. Blanks may occur anywhere on the specifications.

Variable list

- Each variable number must be preceded by a V.
- Variables may be expressed singly (separated by a comma), in ranges (separated by a dash), or as a combination of both (V1, V2, V10-V20).
- The variables may be defined in any order.
- All the variables grouped together in one expression must have the same field width (e.g. for 'V2, V3=10-20' V2 and V3 must both have the same field width defined in the dictionary).
- The variables to be checked may be alphabetic or numeric.

Valid (=) or invalid (<>)

- An = sign indicates that the code values which follow are the valid codes for the variables specified. All other codes will be documented as errors.
- <> (not equal) indicates that the codes which follow are invalid. All cases having these codes for the variables specified will be documented as errors.

List of code values

- Codes may be expressed singly (separated by a comma), in ranges (separated by a dash), or as a combination of both.
- For numeric variables, leading zeros do not have to be entered (e.g. V1=1-10), but remember that several variables being checked for common codes must all have the same field width defined in the dictionary.
- For data with decimal places, do not enter the decimal point in the value, but give the value which accurately reflects the number assuming implied decimal places, e.g. the number 2 with one decimal place should be given as '20'.
- For alphabetic values, trailing blanks do not have to be entered; they are added by the program to match variable width.
- To define a blank or to specify a value containing embedded blanks, enclose the value in primes (e.g. V10='NEW YORK','WASHINGTON',' ').
- Code values may be defined in any order.

Notes.

1) If two different specifications are given for the same variable, only the last one is used.

2) Code specifications for a variable override use of code label records from the dictionary for the variables provided with VARS parameter.

12.7 Restrictions

1. The maximum number of ID variables is 20.
2. The maximum number of distinct codes which can be given on the code specifications is 4000. This restriction can be overcame using ranges of codes since a range of codes counts as only 2 codes.

12.8 Examples

Example 1. Check for illegal codes in qualitative variables and out-of-range values in quantitative variables; the only valid codes for variables V10, V12 and V21 through V25 are 1 to 5 and 9; code 9998 is illegal for variable V35; codes 0 and 8 are illegal for variables V41, V44, V46; variables V71 to V77 should have values within the range 0 to 100, or 999; cases are identified by variables V1, V2 and V4; code values from the dictionary are not used.

```
$RUN CHECK
$FILES
PRINT  = CHECK1.LST
DICTIN = STUDY1.DIC                    input Dictionary file
DATAIN = STUDY1.DAT                    input Data file
$SETUP
JOB TO SCAN FOR ILLEGAL CODES AND OUT-OF-RANGE VALUES
IDVARS=(V1,V2,V4)
V10,V12,V21-V25=1-5,9
V35<>9998
V41,V44,V46<>0,8
V71-V77=0-100,999
```

Example 2. Check for code validity only for a subset of cases (when variable V21 is equal 2 or 3 and variable V25 is equal 1); valid codes for some variables are taken from dictionary C-records; in addition, a code specification is given for variable V48; cases are identified by variable V1.

```
$RUN CHECK
$FILES
DICTIN = STUDY2.DIC                    input Dictionary file
DATAIN = STUDY2.DAT                    input Data file
PRINT  = CHECK.PRT
$SETUP
INCLUDE V21=2,3 AND V25=1
JOB TO SCAN FOR ILLEGAL CODES
IDVARS=V1  VARS=(V18-V28,V36-V41)
V48=15-45,99
```

Chapter 13

Checking of Consistency (CONCHECK)

13.1 General Description

CONCHECK used in conjunction with IDAMS Recode statements provides a consistency check capability to test for illegal relationships between values of different variables. Condition statements in the CONCHECK setup are used to name each check and to indicate which variables are to be listed in the event of an error.

The consistency checks are defined through Recode by testing a logical relationship and then setting the value of a result variable to a value 1 if the relationship is not satisfied, e.g. if V3 cannot logically take the value 9 when V2 takes the value 3 then the following Recode statement can be used:

IF V2 EQ 3 AND V3 EQ 9 THEN R100=1 ELSE R100=0

When an inconsistency is detected in a case, values of specified ID variables for the case are printed. In addition, the values for a set of variables, defined with parameter VARS, are printed. This set is used to get an overall picture of the case in order to more easily detect the reason for the inconsistency and to make sure that a correction for one inconsistency will not cause another. For each consistency condition that fails, a separate set of variables, normally consisting of the particular variables being checked, can be printed along with the number and name of the condition.

13.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases for checking. Variables to be listed when inconsistencies occur are specified with the parameter VARS (for the case) or CVARS (for an individual condition).

Transforming data. Recode statements are used to express the required consistency checks.

Treatment of missing data. CONCHECK makes no distinction between substantive data and missing data values; all data are treated the same.

13.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Inconsistencies. For each case containing an inconsistency, one line of identification is printed consisting of the case sequence number and, optionally, the values of specified ID variables. This is followed by the values of the variables specified with the VARS parameter.

For each individual inconsistency detected in a case, the number and name of the corresponding condition and the values of the variables specified on the condition statement are printed.

Error statistics. At the end of the execution, a summary table is printed giving the number of cases processed, the number of cases containing at least one inconsistency and, for each consistency condition, its number and name, and the number of cases failing the test.

13.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. Numeric or alphabetic variables can be used.

13.5 Setup Structure

```
$RUN CONCHECK

$FILES
     File specifications

$RECODE (optional)
     Recode statements expressing inconsistencies

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Condition statements

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default IDAMS.LST)
```

13.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V1=1
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  TESTING FOR INCONSISTENCIES IN NORTH REGION
   ```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  IDVARS=(V1,V3-V4)  MAXERR=50
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MAXERR=**999**/n
The maximum number of inconsistencies to be printed before CONCHECK will stop.

IDVARS=(variable list)
Up to 5 variables whose values will be listed to identify cases with inconsistencies.
Default: Case sequential number is printed.

VARS=(variable list)
Variables to be listed for any case which has at least one error.

FILLCHAR='string'
Up to 8 characters used to separate variables when listing inconsistencies.
Default: 2 spaces.

PRINT=(CDICT/DICT, VNAMES)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
VNAM	Print the first 6 characters of variable names instead of variable numbers when listing values of variables for inconsistent cases.

4. **Condition statements** (at least one must be given). One condition statement is supplied for each consistency to be tested giving a reference to the corresponding Recode statements, a name for the test and the variables whose values are to be listed when the test fails.

The coding rules are the same as for parameters. Each condition statement must begin on a new line.

```
Example:  TEST=R3  CVARS=(V34,V36,V52)  -
            CNAME='AGE, SEX AND PREGNANCY STATUS'
```

TEST=variable number
Variable for which a non zero value indicates that a consistency check failed.
No default.

CVARS=(variable list)
List of variables whose values will be listed when this inconsistency is encountered.
Default: Only variables specified with IDVARS and VARS will be listed.

CNUM=n
Condition number.
Default: Condition sequence number.

CNAME='string'
Name for this condition, up to 40 characters.
Default: No name.

13.7 Restrictions

1. Only the first 4 characters of alphabetic variables are printed.
2. Condition names may not be more than 40 characters long.
3. Maximum number of ID variables is 5.
4. Maximum number of variables listed for each case in error (VARS list) is 20.
5. Maximum number of variables listed for each condition (CVARS list) is 20.

13.8 Examples

Example 1. Test the relationship between V6 and V7 and between V20 and V21; the identification variables V2 and V3 should be printed for each case with an error along with the values of key variables V8-V10; names of variables should be printed.

```
$RUN CONCHECK
$FILES
PRINT  = CONCH1.LST
DICTIN = MY.DIC                      input Dictionary file
DATAIN = MY.DAT                      input Data file
$RECODE
  R1=0
  R2=0
  IF V5 INLIST(1-5,8) AND V7 EQ 2 THEN R1=1
  IF V20 LE 3 AND V21 EQ 5 OR V20 EQ 8 AND V21 EQ 7 OR V20 EQ V21 THEN R2=1
$SETUP
TESTING FOR 2 INCONSISTENCIES
PRINT=VNAMES  IDVARS=(V2,V3)  VARS=(V8-V10)
TEST=R1  CNAME='1st Inconsistency' CVARS=(V5,V7)
TEST=R2  CNAME='2nd Inconsistency' CVARS=(V20,V21)
```

Example 2. Test 5 conditions in part 2 of a questionnaire; tests are numbered starting at 201; all variables from part 2 should be listed for each questionnaire with an error, along with key variables from part 1 (V5-V10); in addition, particular variables used in tests should be listed again for each test that fails. Note the use of the Recode SELECT function to initialize the corresponding result variables to 0.

```
$RUN CONCHECK
$FILES
DICTIN = MY.DIC                      input Dictionary file
DATAIN = MY.DAT                      input Data file
$SETUP
PART 2 OF CONSISTENCY CHECKING
MAXERR=400  IDVARS=(V1,V3)  VARS=(V5-V10,V200-V231)
TEST=R1  CNUM=201  CVARS=(V203-V205)
TEST=R2  CNUM=202  CVARS=(V203,V210-V212)
TEST=R3  CNUM=203  CVARS=(V214,V215)
TEST=R4  CNUM=204  CVARS=(V222-V226)
TEST=R5  CNUM=205  CVARS=(V229,V230)
$RECODE
      R900=1
A     SELECT (FROM=(R1-R5), BY R900) = 0
      IF R900 LT 5 THEN R900=R900+1 AND GO TO A
      IF V203 IN(1-5,17,20-25) AND V204 EQ 3 OR V205 EQ 'M' THEN R1=1
      IF V203 GT 6 AND MDATA(V210,V211,V212) THEN R2=1
      IF 2*TRUNC(V214/2) EQ V214 OR V215 EQ 0 THEN R3=1
      IF COUNT(1,V222-V226) LT 2 THEN R4=1
      IF MDATA(V229) AND NOT MDATA(V230) THEN R5=1
```

Chapter 14

Checking the Merging of Records (MERCHECK)

14.1 General Description

The MERCHECK program detects and corrects merge errors (missing, duplicate or invalid records) in a data file containing multiple records per case. It outputs a file containing equal numbers of records per case by padding in missing records and deleting duplicate and invalid records. Although originally written for checking card-image data, the input data record length may be any value up to 128. Since all other IDAMS programs assume that each case in a data file has exactly the same number of records, using MERCHECK is an essential first checking step for all data files which have more than one record per case.

Program operation. The user supplies a set of Record descriptions defining the permissible record types. While processing the data, the program reads into a work area all the contiguous input data records it finds which have identical case ID values. These records are compared one by one with the defined record types, and an output case is constructed. Records are padded, deleted, reordered, etc., as needed. The data case is then transferred to the output file, and the program returns to read the set of input records for the next case. The results document the corrections of the input data performed by the program.

Case and record identification. MERCHECK requires that the case ID is in the same position for all records. Case ID fields may be located in non-contiguous columns and may be composed of any characters. Record types are identified by a single record ID field (of 1-5 columns) which may be composed of any character except a blank. A sketch of a data file with two record types follows. The intervening periods stand for data or blank fields.

```
...SE23...01..............10......
...SE23...01..............12......
...SE23...02..............10......
...SE23...02..............12......
...SE24...01..............10......
...SE24...01..............12......

 first    second           record ID
 case ID  case ID          field
 field    field
```

In the example, there are 2 types of record for each case, identified by a 10 or 12 in columns 28, 29. The case ID consists of two non-contiguous fields, columns 4-7 and columns 11-12. Thus “SE2301” is a case ID, as are “SE2302” and “SE2401”.

Eliminating invalid records. An input data record containing a record ID not defined by the Record descriptions, known as an “extra” record, is optionally printed but never transmitted to the output file. In addition, there are two options for eliminating other types of invalid records.

- Records which do not contain a specified constant are rejected. (See the parameters CONSTANT, CLOCATION, and MAXNOCONSTANT).

- The user may supply the case ID value of the first valid data case. All records containing a case ID value less than the one specified are rejected. (See the parameter BEGINID).

Options to handle cases with missing records. The user must select, using the parameter DELETE, one of the three possible ways to handle incomplete cases.

1. DELETE=ANYMISSING. A case is not output if one or more of its record types is missing.

2. DELETE=ALLMISSING. A case is not output if not a single valid record ID is found for a particular case ID.

3. DELETE=NEVER. The program never excludes from the output file a case missing one or more records. Instead, it constructs a record for each missing record type and "pads" its contents with blanks or user-supplied values. See the PADCH parameter and the PAD parameter on the Record descriptions. Padding takes place in column locations other than the case and record ID fields. The appropriate case and record ID's are always inserted by the program.

Options to handle cases with duplicate records. A duplicate record is one having the same case and record ID's as another record regardless of the rest of the contents of the two records. The user specifies which duplicate is to be kept if there is more than one input record bearing the same case and record ID's. For example, the option DUPKEEP=1 causes the program to retain the first record and to discard any others. The case is not transferred to the output file if fewer than n duplicates are found (where DUPKEEP=n) i.e. to delete cases with duplicate records, specify a large value for n. Caution: It may happen that records with duplicate ID's do not contain the same data. It is up to the user to determine the appropriateness of the record that was retained.

Options to handle deleted records. Those input data records which are deleted, i.e. not written to the output file, may be saved in a separate file (see the parameter WRITE).

Selection of record types. MERCHECK allows the user to subset selected record types from a more comprehensive input data file. Simply include only the required ID's in the Record descriptions, and choose an appropriate error printing option (EXTRAS=n or PRINT=ERRORS, for example) and a realistic MAXERR value. Minimizing printed output for cases in error is essential, as nearly every case in the input data file will be reported in error due to records with invalid record ID's (i.e. those not specified on Record descriptions).

Restart capabilities. The parameter BEGINID can be used to restart MERCHECK if a prior execution terminated before all input data were processed. The user must determine the case ID value for the last case output and set BEGINID equal to that value +1. (If termination occurred because the parameter MAXERR was exceeded, the last input record read will appear displayed in the results, and BEGINID should be set to the case ID of that record).

Note. MERCHECK is intended for checking data files with multiple records per case and there must be a record ID entered in each record. MERCHECK could theoretically be used for eliminating duplicate records and records without a particular constant for data files with a single record per case. This however can only be done if each data record contains a constant value which can be treated as the record ID. This operation is better performed by the SUBSET program, using a filter to exclude records without a constant and the DUPLICATE=DELETE option to eliminate duplicates. (See write-up for SUBSET).

14.2 Standard IDAMS Features

Case and variable selection. Except as defined above, not available for this program.

Transforming data and missing data. These options do not apply in MERCHECK.

14.3 Results

Error cases. The full report with the documentation of each error case has three parts: an error summary, the records not transferred to the output (bad records), and the case as it appears in the output file (good records). See below for more details of these components. For data with a large number of record types and with many cases in error, the report for error cases can be costly and, for some jobs, quite unnecessary. The amount of report needed depends on how much a user knows about the data, as well as the ability to correct or double-check the errors. For instance, if a user expects considerable padding to occur, but virtually no duplicate or invalid records, it may be sufficient to have only the error summary printed and to specify that cases with errors (if any) be saved (see the option WRITE=BADRECS) and listed later. Various controls on the quantity of results are possible with the parameters PRINT, EXTRAS, DUPS, and PADS.

Error cases: error summary. The error summary consists of an identification of the error case (case count or case ID) and any of three messages about the errors which occurred. The sequential case count does not account for records or cases eliminated because they appear before the beginning ID or lack the required constant. The case ID is taken from the case ID field(s) as specified by the IDLOC parameter.

The 3 kinds of errors are reported, namely:

1. invalid record types,
2. cases with missing records,
3. cases with duplicate records.

Error cases: bad records. There are the invalid and duplicate records as well as all records for cases which have been rejected because of missing records. They are printed in the order that they appear in the input file.

Error cases: good records. If a case is kept after an error has been encountered, the actual records written to the output file, including any padding records, are listed.

Records occurring before the one with BEGINID. These are optionally printed. See the parameter PRINT=LOWID.

Records out of sort order. These are normally printed although results can be suppressed. See the parameter PRINT=NOSORT.

Records without the specified constant. Any record which does not contain the user specified constant in the correct columns is printed. This report can be suppressed. See the parameter PRINT=NOCONSTANT.

Execution statistics. At the end of the report the total number of missing records, invalid records and duplicate records, and the total number of cases which were read, written, deleted and containing errors are printed.

14.4 Output Data

The output data is a file with the same record length as the input data and equal number of records per case. Each case contains one each of the record types specified on the Record descriptions.

14.5 Input Data

The input consists of a file of fixed length data records normally sorted by case ID and record ID within case. The record length may not exceed 128.

14.6 Setup Structure

```
$RUN MERCHECK

$FILES
     File specifications

$SETUP
     1. Label
     2. Parameters
     3. Record descriptions (repeated as required)

$DATA (conditional)
     Data


Files:
FT02       rejected records ("bad case" records)
           when WRITE=BADRECS specified
DATAxxxx   input data (omit if $DATA used)
DATAyyyy   output data (good cases)
PRINT      results (default  IDAMS.LST)
```

14.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  CHECKING THE MERGE OF RECORDS IN STUDY 95 DATA
   ```

2. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  MAXE=25 RECORDS=8 IDLOC=(1,5)
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Data file.
   Default ddname: DATAIN.

   MAXCASES=n
   The maximum number of cases to be used from the input file.
   Default: All cases will be used.

   MAXERR=**10**/n
   Maximum number of cases with errors. When n + 1 error cases occur, execution terminates. Cases before the BEGINID, those out of sort order, and records without the constant do not count as error cases. Error cases are those with invalid, duplicate, or missing records.

   OUTFILE=**OUT**/yyyy
   A 1-4 character ddname suffix for the output Data file.
   Default ddname: DATAOUT.

RECORDS=**2**/n
The number of records per case (as defined on the Record descriptions).

IDLOC=(s1,e1, s2,e2, ...)
Starting and ending columns of 1-5 case identification fields. At least one must be given. If there is more than one case ID field, then they must be specified in the order in which the input data are sorted.
No default.

BEGINID='case id'
Lowest valid case ID value at which program begins processing: 1 to 40 characters enclosed in primes if contain any non-alphanumeric characters. If multiple case ID fields are used, the value should be the concatenation of the individual case ID's supplied in sort order.
Default: Blanks.

NOSORT=**0**/n
The maximum number of cases out of sort order tolerated by the program. When n + 1 cases out of order occur, execution terminates.

DELETE=**NEVER**/ANYMISSING/ALLMISSING
Specifies under what conditions with respect to missing records a case is to be deleted.

NEVE	Never reject a case due to missing records. If any or all of the records are missing, the program will pad (with blanks or user-supplied values) all records which are missing and reject any records with invalid record ID's before outputting the case.
ANYM	Do not output any case in which one or more records is missing, i.e. no incomplete case is to be output.
ALLM	Do not output any case in which there are no valid records, i.e. when all records for a case have invalid record ID's.

PADCH=x
Character to be used on padded records. Non-alphanumeric character must be enclosed in primes. See also Record descriptions for more detailed padding values.
Default: Blank.

DUPKEEP=**1**/n
Specifies (for duplicate data records) that the n-th duplicate encountered is to be kept. If fewer than n duplicates are found, the case in which they occur is deleted (even if DELETE=NEVER is specified).

WRITE=BADRECS
Create a file of the rejected (bad case) records.

CONSTANT=value
Value of a constant. Must be enclosed in primes if it contains non-alphanumeric characters. Any input data record without the constant is rejected. The location of the constant must be the same across all input records regardless of record type.

CLOCATION=(s, e)
(Supplied only if CONSTANT is used). Location of the constant field.

s	Starting column of constant's field on each record.
e	Ending column of constant's field on each record.

MAXNOCONSTANT=**0**/n
(Supplied only if CONSTANT is used). Maximum number of records without the constant tolerated by the program. When n + 1 records without the constant are encountered, MERCHECK terminates execution.

PRINT=(**CONSTANT**/NOCONSTANT, **SORT**/NOSORT, **ERRORS**/NOERRORS, LOWID, BADRECS, GOODRECS)

CONS Print records without specified constant.
NOCO Do not print records without the constant.
SORT Print a 3-line notice for cases out of sort order.
NOSO Do not print cases out of sort order.
LOWI Print all records with case ID lower than the one specified with BEGINID.

The following print options refer to the report of cases with errors (i.e. missing, invalid, or duplicate records).

ERRO Print error summary for each case with an error.
NOER Do not print error summary for cases with errors.
BADR Print rejected (bad) records for cases with errors.
GOOD Print kept (good) records for cases with errors.

EXTRAS=**0**/n
DUPS=**0**/n
PADS=**0**/n

If a case has fewer than n invalid (extra/duplicate/padded) records and no other errors, no report will occur for the case. Thus, a case with only 2 invalid records and no missing or duplicate records would not generate report if EXTRAS=3, but would print according to the PRINT specification if it also had 1 missing record.
Default: All error cases will be printed according to PRINT specification.

3. **Record descriptions** (mandatory: one for each type of record to be selected for output). The coding rules are the same as for parameters. Each record description must begin on a new line.

```
Example:  RECID=21   RIDLOC=1
          RECID=3    RIDLOC=2   PAD='43599-
          999998889999999881119'
```

RECID=xxxxx

A 1-5 non-blank character record type code. Must be enclosed in primes if it contains lower case characters.
No default.

RIDLOC=s

Starting column of record ID field.
No default.

PAD='xxx....'

Pad values to be used when padding a record of this type. The string of values must be enclosed by primes if it contains non-alphanumeric characters. The first character will be put in column 1 of the output padded record, etc. To continue on a subsequent line, enter a dash. If the length of the string is less than the record length, then the rest of the string is filled on the right with the PADCH specified on the parameter statement.
Default: PADCH is used for entire string.
Note: The correct case ID and record ID are automatically inserted into the padded record in the correct positions.

14.8 Restrictions

1. Maximum record length of input data records is 128.
2. Maximum number of output records per case is 50.
3. The program reserves work space for a maximum of 60 records with identical case ID value. Included in the count are invalid, duplicate, and valid records, and also records which are padded by the program. MERCHECK terminates execution if more than 60 records with identical case ID values occur in the work area.

4. Maximum combined length of the individual case ID fields is 40 characters.
5. Maximum length of the record ID field is 5 contiguous non-blank characters.
6. Maximum length of a constant to be checked for is 12 characters.
7. Maximum number of case ID fields is 5.

14.9 Examples

Example 1. Check the merge of three records per case which have record types 1, 2 and 3 respectively; missing records are padded: records 1 and 2 are padded with blanks, record 3 is padded with a copy of the values given with the PAD parameter; cases with no valid records (when all records for a case have invalid record types) are written to the file BAD; cases with up to four duplicate records are also written to the file BAD (if a case has 5 or more duplicates of a particular record type, then it is kept as a good case using the 5th of the duplicates and eliminating the others).

```
$RUN MERCHECK
$FILES
PRINT   = MERCH1.LST
FT02    = \DEMO\BAD           file for output bad cases
DATAIN  = \DEMO\DATA1         input Data file
DATAOUT = \DEMO\DATA2         output Data file (with only good cases)
$SETUP
CHECKING THE MERGE OF DATA
IDLO=(1,3,5,6,10,10)  RECO=3  DELE=ALLM  DUPK=5  WRITE=BADRECS  MAXE=200
RECID=1  RIDLOC=12
RECID=2  RIDLOC=12
RECID=3  RIDLOC=12   PAD='9999999999-
939999999999999999999999999999999999999999999999999999999999999999'
```

Example 2. Check data, deleting all cases with missing records and eliminating cases which do not belong to the study; Data file contains two records per case; cases with duplicate records are kept (dropping all except the first of a set of duplicate records); there is a record type TT in columns 4 and 5 of one record and one of AB in columns 7 and 8 of the other; the study ID, HST, should appear in columns 124-126 of each record.

```
$RUN MERCHECK
$FILES
FT02    = BAD                 file for output bad cases
DATAIN  = DATA  RECL=126      input Ddata file
DATAOUT = GOOD                output Data file (with only good cases)
$SETUP
CHECKING THE MERGE OF DATA
IDLO=(1,3)  RECO=2  WRITE=BADRECS  MAXE=20  -
   CONS=HST  CLOC=(124,126)
RECID=TT  RIDLOC=4
RECID=AB  RIDLOC=7
```

Chapter 15

Correcting Data (CORRECT)

15.1 General Description

CORRECT provides correction facilities for data in an IDAMS dataset. Individual variable values in specified cases may be corrected or entire cases deleted.

CORRECT is useful for correcting errors in individual variables for specific cases as detected for example by BUILD, CHECK or CONCHECK. The preparation of update instructions is easy. Checks are made for compatibility between the data and the correction and good documentation is printed describing all the corrections made.

Program operation. CORRECT first reads the dictionary and stores the information about all the variables in the dataset. Each data correction instruction is then processed. After an instruction is read, CORRECT reads the data file copying cases until the case identified in the instruction is encountered. CORRECT executes the instruction, listing the case, or revising values for selected variables and outputting the case, or deleting the case from the output as appropriate. When all instructions are exhausted, the remaining data cases (if any) are copied to the output, and execution terminates normally. If errors in the sort order of the correction instructions or data cases occur and also if there are syntax errors on the correction instructions, CORRECT documents the situation in the results and continues with the next instruction.

Variable correction. The user specifies the case identification followed by the variable numbers of the variables to be corrected together with their new values. Both numeric (integer or decimal valued) and alphabetic variables can be corrected.

Correcting case ID variables. If an ID field is to be corrected, normally the sort order will be affected and the parameter CKSORT=NO should therefore be specified. If the ID variable contains erroneous non-numeric characters, then enclose its value in primes on the correction instruction.

Case deletion. The user can delete a case from the data file by specifying case identification information and the word "DELETE".

Case listing. The user can choose to have a particular data case listed by specifying case identification information and the word "LIST".

15.2 Standard IDAMS Features

Case and variable selection. One may select a subset of cases to be processed and output by including a standard filter. Selection of variables is inappropriate.

Transforming data. Recode statements may not be used.

Treatment of missing data. CORRECT makes no distinction between substantive data and missing data values; the concept does not apply to the program operation.

15.3 Results

Input dictionary. (Optional: see the parameter PRINT). Dictionary records for all variables are printed, not just for those being corrected.

Listing of the correction instructions. Correction instructions are always listed. With each correction the program also optionally lists: (1) input data records, (2) deleted records, or (3) corrected records (see PRINT parameter).

15.4 Output Dataset

A copy of the dictionary is always output. If it is not required, the DICTOUT file definition can be omitted. The data are always copied to the output, even if there are no corrections or deletions.

15.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. Normally, CORRECT expects the data cases to be sorted in ascending order on values of their case ID variables. The user can, however, indicate (via the parameter CKSORT) that the cases are not in ascending order. This option should be used with caution: the order of the correction instructions must exactly match the order of the data in the file.

15.6 Setup Structure

```
$RUN CORRECT

$FILES
     File specifications

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Correction instructions (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default  IDAMS.LST)
```

15.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  INCLUDE V1=10,20,30 AND V12=1,3,7
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  CORRECTION OF ALPHA CODES IN 1968 ELECTION
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  PRINT=CORRECTIONS, IDVARS=V4
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input dictionary and data files.
Default ddnames: DICTIN, DATAIN.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file. If MAXC=0, all correction instructions will be checked for syntax errors but no data processed.
Default: All cases will be used.

IDVARS=(variable list)
Up to 5 variable numbers for the case identification fields. If more than one case ID field is specified, the variable numbers must be given in major to minor sort field order.
No default.

CKSORT=**YES**/NO
Indicates whether the data cases will have their case ID field(s) checked for ascending sequential ordering. The execution terminates if a case out of order is detected.

OUTFILE=**OUT**/yyyy
A 1-4 character ddname suffix for the output dictionary and data files.
Default ddnames: DICTOUT, DATAOUT.

PRINT=(DELETIONS, CORRECTIONS, CDICT/DICT)

DELE	List those cases for which the delete option is specified in correction instructions.
CORR	List corrected cases.
CDIC	Print the input dictionary for all variables with C-records if any.
DICT	Print the input dictionary without C-records.

4. **Correction instructions.** These statements indicate which of the listing, deletion, or correction options are to be applied and for which cases.

```
Examples:

ID=1026,V5=9,-          (For the case with ID "1026" change the
    V6=22                value of V5 to 9 and the value of V6 to 22)
ID='JOHN DOE',DELETE    (Delete the case with ID "JOHN DOE" from the output)
ID=091,3,LIST           (List the case with ID "091", "3")
ID=023,16,V8='DON_T',-  (Change V8 to DON'T and V9 to TEACH,RES)
V9='TEACH|RES'
```

Rules for coding

Each correction instruction must start on a new line. To continue to another line, break after the comma at the end of a complete variable correction and enter a dash. As many continuation lines may be used as necessary. Blanks may occur anywhere on the instructions.

The correction instructions must be ordered in exactly the same relative sequence by case ID values as the data cases.

Case ID values

- The case to be corrected is identified using the keyword "ID=" followed by the value(s) of the ID variable(s).
- The list of values on the instruction is not enclosed in parentheses.
- Each value, including the last, must be followed by a comma, and the order of the values should correspond to the order of the variables in the list of ID variables specified with the IDVARS parameter.
- The number of digits or characters in a value must equal the width of the variable as stated in the dictionary, i.e. leading zeros may need to be included.
- Values containing non-numeric characters should be enclosed in primes, e.g. ID=9,'PAM'.

Type of instruction

The case identification is followed either by the word "LIST", by the word "DELETE", or by a string of variable corrections.

Variable corrections

- A variable correction consists of a variable number preceded by a "V" and followed by an "=" and the correct value, e.g. V3=4.
- Variable corrections for different variables for the same case are separated by commas.
- Correction values for numeric variables may be specified without leading zeros.
- If the variable includes decimal places, the decimal point may be entered, but is not written to the output file. The digits are aligned according to the number of decimal places indicated in the dictionary and excess decimal digits are rounded.
- If the value contains non-numeric characters it must be enclosed in primes. An embedded comma must be represented as a vertical bar and an embedded prime must be represented as an underscore; the program will convert the vertical bar and underscore to the comma and prime respectively, e.g. v8='Don_t'.
- Correction values for alphabetic variables must match the variable width. If the correction value contains blanks or lower case characters it should be enclosed in primes.

15.8 Restriction

The maximum number of case ID variables is 5.

15.9 Example

Correction of data file; both numeric and alphabetic variables are to be corrected, and two cases are to be deleted; cases are identified by variables V1, V2 and V5; the dictionary is not changed, and therefore an output dictionary is not needed.

```
$RUN CORRECT
$FILES
PRINT   = CORRECT1.LST
DICTIN  = DATA1.DIC              input Dictionary file
DATAIN  = DATA1.DAT              input Data file
DICTOUT = DATA2.DIC              output Dictionary file (same as input)
DATAOUT = DATA2.DAT              output Data file (corrected)
$SETUP
CORRECTING A DATA FILE
IDVARS=(V1,V2,V5)
ID=311,01,21,V12='JOHN MILLER'
ID=311,05,41,DELETE
ID=557,11,32,V58=199,V76=2,V90=155
ID=559,11,35,V12='AGATA CHRISTI',V13='F'
ID=657,31,11,V58=100,V77=4,V90=105,V36=999999,V37=999999,V38=999999, -
   V41=98,V44=99
ID=711,15,11,DELETE
```

Chapter 16

Importing/Exporting Data (IMPEX)

16.1 General Description

The IMPEX program performs import/export of data in free or DIF format, and import/export of matrices in free format. In a free format file, fields may be separated with space, tabulator, comma, semicolon or any character defined by the user. Decimal point or comma can be used in decimal notation. Imported/exported Data file may contain variable numbers and/or variable names as column headings. Imported/exported matrix file may contain variable numbers/code values and/or variable names/code labels as column/row headings.

Data import. The program creates a new IDAMS dataset from an existing free or DIF (format for data interchange developed by Software Arts Products Corp.,) format ASCII data file and from an IDAMS dictionary. The input dictionary defines how the fields of the input data file must be transferred into the output IDAMS dataset.

Data export. The program creates a new ASCII data file containing variables from an existing IDAMS dataset and new variables defined by IDAMS Recode statements. The exported file may be of free or DIF format.

Matrix import. The program creates an IDAMS Matrix file from a free format ASCII file containing a lower triangle of a square matrix or a rectangular matrix.

Matrix export. The program creates an ASCII file containing all matrices stored in an IDAMS Matrix file. For matrix export, only free format is available.

16.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data when data export is requested. Also in data export, variables are selected through the parameter OUTVARS.

Transforming data. Recode statements may be used in data export.

Treatment of missing data. No missing data checks are made on data values except through the use of Recode statements in data export. In data import, empty fields (empty fields between consecutive delimiters) are replaced with the first missing data code or with a field of 9's if the first missing data code is not defined.

16.3 Results

Data Import

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, for all variables included in the input dictionary.

Input column labels and codes. (Optional: see the parameters PRINT and EXPORT/IMPORT). Column labels and column codes are printed (unformatted) as they are read from the input file.

Input data. (Optional: see the parameter PRINT). Unformatted input data lines are printed for all cases exactly as they are read from the input data file.

Output dictionary. (Optional: see the parameter PRINT).

Output data. (Optional: see the parameter PRINT). Values for all cases and for all variables are given, 10 values per line, in the same order as input data lines.

Data Export

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Output data. (Optional: see the parameter PRINT). Values for all cases for each V- or R-variable are given, 10 values per line. For alphabetic variables, only the first 10 characters are printed.

Matrix Import

Input matrix. (Optional: see the parameter PRINT). A matrix contained in the input ASCII file is printed with or without column labels and column codes.

Matrix Export

Input matrices. (Optional: see the parameter PRINT). Matrices contained in the input IDAMS matrix file are printed with or without variable descriptor records or code label records.

16.4 Output Files

Import

The output is either an IDAMS dataset or an IDAMS matrix depending on whether data or matrix import is requested.

In the case of an IDAMS dataset, values of the numeric variables are edited according to IDAMS rules (see the "Data in IDAMS" chapter).

Empty numerical fields (i.e. empty strings between delimiter characters) in a free format input file are replaced with the corresponding first missing data code or with 9's if the first missing data code is not defined.

Export

The output is an ASCII file, the content of which varies according to the export requirements.

Data in DIF format. This is a file with standard "Header" and "Data" sections. Vectors correspond to IDAMS variables, and "TUPLES" to cases. In addition to the required header items, LABEL (a standard optional item) is used to export variable names. In the Data section, the Value Indicator "V" is always used for numeric values. A decimal point or comma is used in decimal notation if the number of decimals defined in the dictionary is greater than zero.

Data in free format. This is a file in which variable values are separated by a delimiter (see the parameters WITH and DELCHAR) and cases are separated additionally by carriage return plus line feed characters. For numeric variable values, a decimal point or comma (see the parameter DECIMALS) is included if the number of decimals defined in the dictionary is greater than zero. Alphabetic variable values may be enclosed in primes or quotes, or not enclosed in any special characters (see the parameter STRINGS).

Matrix in free format. The format of matrices output by IMPEX is the same as the format required for imported matrices (see "Matrix Import" in the "Input Files" section below). The only difference is that additional delimiter characters are inserted to ensure correct positioning of column and row labels in a spreadsheet package.

16.5 Input Files

Data Import

For data import, the input is:

- an ASCII file containing a free format data array in which fields are separated with a delimiter, and an IDAMS dictionary which defines how to transfer data into an IDAMS dataset (all fields have to be described in the input dictionary);
- a DIF format data file, and also an IDAMS dictionary.

The input files may also contain dictionary information. For free format files, this means that column labels and column codes (which correspond to variable names and variable numbers) are supplied with the data array as the first rows in the array. Both labels and codes are optional. If provided, column labels override variable names from the input dictionary, and they are inserted in the output dictionary. They may be enclosed in special characters (see the parameter STRINGS). Column codes are used only to perform a check against variable numbers from the input dictionary. For DIF format files, column labels appear as LABEL items in the Header section. Column codes can be present as the first row in the data array.

Matrix Import

The input is always a free format ASCII file in which numerical values/strings of characters are separated with a delimiter. Empty fields (i.e. empty strings between delimiter characters) are skipped. Each file may contain only one matrix to import.

The input matrix file may optionally provide dictionary information consisting of a series of strings for labelling columns/rows of the matrix and the corresponding codes. If provided, they must follow the syntax given below (which is different for rectangular and square matrices).

Rectangular matrix

This is an ASCII file containing a free format rectangular array of values; dictionary information may be optionally included.

Example.

```
Average salary; Age group; Sex;
Male; Female;
1;2;
20 - 30;1;600;530;
31 - 40;2;650;564;
41 - 60;3;723;618;
```

Format.

1. The first three strings contain, respectively: (1) a description of the matrix contents, (2) the row title ("row variable name"), and (3) the column title ("column variable name"). (Optional).
2. Column labels. (Optional: one label per column of the array of values).
3. Column codes. (Optional: one code per column of the array of values).
4. The array of values. (This may optionally contain one row label and/or code before each row of values).

Note. If row and column labels and/or codes are not present, they are automatically generated for the output IDAMS matrix (labels as R-#0001, R-#0002, ... C-#0001, C-#0002, ... and codes from 1 to the number of rows and columns respectively).

Square matrix

This is an ASCII file containing a lower-left triangle of a matrix (only off-diagonal elements), and optionally vectors of means and standard deviations following the matrix, in free format.

Example.

```
;;Paris;London;Brussels;Madrid; ...
;;1;2;3;4; ...
Paris;1;
London;2;0.55;
Brussels;3;0.45;0.35;
Madrid;4;1.45;2.35;1.15;
. . .
```

Format.

1. Column labels ("variable names"). (Optional: as many labels as columns/rows in the array of values).
2. Column codes ("variable numbers"). (Optional: as many codes as columns/rows in the array of values).
3. The array of values. (This may optionally contain one row label and/or code before each row of values).
4. A vector of means. (Optional).
5. A vector of standard deviations. (Optional).

Note. If labels and/or codes are not present, they are automatically generated for the output IDAMS matrix (labels as V-#0001, V-#0002, ... and codes from 1 to the number of columns/rows).

Data and Matrix Export

Depending on whether data or matrix(ces) are to be exported, the input is either a data file described by an IDAMS dictionary (both numeric and alphabetic variables can be used) or a file of IDAMS square or rectangular matrix(ces).

16.6 Setup Structure

```
$RUN IMPEX

$FILES
     File specifications

$RECODE (optional with data export; unavailable otherwise)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary for data export/import (omit if $DICT used)
DATAxxxx   input data/matrix (omit if $DATA used)
DICTyyyy   output dictionary for data import
DATAyyyy   output data/matrix
PRINT      results (default  IDAMS.LST)
```

16.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution if data export is specified.

   ```
   Example:  EXCLUDE V19=2-3
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  EXPORTING SOCIAL DEVELOPMENT INDICATORS
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  EXPORT=(DATA,NAMES)  FORMAT=DELIMITED WITH=SPACE
   ```

   IMPORT=(**DATA**/MATRIX, NAMES, CODES)

DATA	Data import is requested.
MATR	Matrix import is requested.
NAME	Variable names are included in the Data file to import. Variable names/code labels are included in the Matrix file to import.
CODE	Variable numbers are included in the Data file to import. Variable numbers/code values are included in the Matrix file to import.

EXPORT=(**DATA**/MATRIX, NAMES, CODES)

DATA Data export is requested.
MATR Matrix export is requested.
NAME Variable names are to be exported in the outpur Data file. Variable names/code labels are to be exported in the outpur Matrix file.
CODE Variable numbers are to be exported in the output Data file. Variable numbers/code values are to be exported in the output Matrix file.

Note. No defaults. Either IMPORT or EXPORT (but not both) must be specified.

INFILE=**IN**/xxxx

A 1-4 character ddname suffix for the input file(s):
Data or Matrix file to import (default ddname: DATAIN),
Dictionary and Data files to export data (default ddnames: DICTIN, DATAIN),
IDAMS Matrix file to export (default ddname: DATAIN).

BADDATA=**STOP**/SKIP/MD1/MD2

Treatment of non-numeric import or export data values and "insufficient field width" output values. See "The IDAMS Setup File" chapter.

MAXCASES=n

Applicable only if data import/export is specified.
The maximum number of cases (after filtering) to be used from the input data file.
Default: All cases will be used.

MAXERR=**0**/n

The maximum number of "insufficient field width" errors allowed before execution stops. These errors occur when the value of a variable is too big to fit into the field assigned, e.g. a value of 250 when a field width of 2 has been specified.

OUTFILE=**OUT**/yyyy

A 1-4 character ddname suffix for the output file(s):
Dictionary and Data files obtained by import (default ddnames: DICTOUT, DATAOUT),
IDAMS Matrix file obtained by import (default ddname: DATAOUT),
exported Data or Matrix file (default ddname: DATAOUT).

OUTVARS=(variable list)

Applicable only if data export is specified.
V- and R-variables which are to be exported. The order of the variables in the list is not significant, since they are output in ascending numerical order. All V- and R-variable numbers must be unique.
No default.

MATSIZE=(n,m)

Applicable only if matrix import is specified.
Number of rows and columns of the matrix to import. The program assumes a rectangular matrix if both are specified and a square symmetric matrix if one of them is omitted.
n Number of rows.
m Number of columns.
No default.

FORMAT=**DELIMITED**/DIF

Specifies the input data/matrix format for import, or the output data/matrix format for export.
DELI Data/matrix(ces) is expected to be of free format, in which fields are separated with a delimiter (see below).
DIF Data are expected to be in DIF format.
Note: DIF format is available only for data export or import.

WITH=**SPACE**/TABULATOR/COMMA/SEMICOLON/USER
(Conditional: see FORMAT=DELIMITED).
Specifies the delimiter character to separate fields in free format file.
SPAC Blank character (ASCII code: 32).
TABU Tabulator character (ASCII code: 9).
COMM Comma "," (ASCII code: 44).
SEMI Semicolon ";" (ASCII code: 59).
USER User specified character (see the parameter DELCHAR below).
Note: In importing/exporting DIF files, COMMA is always used as the delimiter character, independently of what is selected.

DELCHAR='x'
(Conditional: see the parameter WITH=USER above).
Defines the character used to separate fields in free format files.
Default: Blank.

DECIMALS=**POINT**/COMMA
Defines the character used in decimal notation.
POIN Point "." (ASCII code: 46).
COMM Comma "," (ASCII code: 44).

STRINGS=**PRIME**/QUOTE/NONE
Defines the character used to enclose character strings.
PRIM Prime.
QUOT Quote.
NONE No special character is used.
Note: In importing/exporting DIF files, QUOTE is always used, independently of what is selected.

NDEC=**2**/n
Number of decimal places to be retained in export.

PRINT=(**DICT**/CDICT/NODICT, DATA)
DICT Print the dictionary without C-records.
CDIC Print the dictionary with C-records if any.
DATA Print data values.

Note:

(a) Dictionary printing options control both input and output dictionary printing.

(b) Data printing option controls output data printing if a data file is exported, and controls both input and output if data import is requested (input is never printed if a DIF format data file is imported).

(c) For matrices, the input matrix is printed whenever data printing is specified.

16.8 Restrictions

1. The maximum number of R-variables that can be exported is 250.
2. The maximum number of variables that can be used in one execution (including variables used only in Recode statements) is 500.
3. The maximum number of matrix rows is 100.
4. The maximum number of matrix columns is 100.
5. The maximum number of matrix cells is 1000.

16.9 Examples

Example 1. Selected variables from the input dataset are transferred to the output file along with two new variables; data are output in free format with values separated by a semicolon; commas will be used in decimal notation while alphabetic variable values will be enclosed in quotes; variable names and variable numbers will be included in the output data file.

```
$RUN IMPEX
$FILES
PRINT   = EXPDAT.LST
DICTIN  = OLD.DIC                       input Dictionary file
DATAIN  = OLD.DAT                       input Data file
DATAOUT = EXPORTED.DAT                  exported Data file
$SETUP
EXPORTING IDAMS FIXED FORMAT DATA TO FREE FORMAT DATA
EXPORT=(DATA,NAMES,CODES)  BADD=MD1  MAXERR=20  -
   OUTVARS=(V1-V20,V33,V45-V50,R105,R122)  -
   FORMAT=DELIM  WITH=SEMI  DECIM=COMMA  STRINGS=QUOTE
$RECODE
   R105=BRAC(V5,15-25=1,<36=2,<46=3,<56=4,<66=5,<90=6,ELSE=9)
   MDCODES R105(9)
   NAME R105'GROUPS OF AGE'
   IF MDATA(V22) THEN R122=99.9 ELSE R122=V22/3
   MDCODES R122(99.9)
   NAME R122'NO ARTICLES PER YEAR'
```

Example 2. DIF format data are imported to IDAMS; column labels and column codes are included in the input data file, and commas are used in decimal notation.

```
$RUN IMPEX
$FILES
PRINT   = IMPDAT.LST
DICTIN  = IDA.DIC             Dictionary file describing data to be imported
DATAIN  = IMPORTED.DAT        Data file to be imported
DICTOUT = IDAFORM.DIC         output Dictionary file
DATAOUT = IDAFORM.DAT         output Data file
$SETUP
IMPORTING DIF FORMAT DATA TO IDAMS FIXED FORMAT DATA
IMPORT=(DATA,NAMES,CODES)  BADD=MD1  MAXERR=20  -
   FORMAT=DIF  DECIM=COMMA
```

Example 3. A set of rectangular matrices created by the TABLES program is exported; values will be separated by a semicolon and commas will be used in decimal notation; column and row labels and codes will be included in the output matrix file; input matrices are printed.

```
$RUN IMPEX
$FILES
PRINT   = EXPMAT.LST
DATAIN  = TABLES.MAT                    file with rectangular matrices
DATAOUT = EXPORTED.MAT                  file with exported matrices
$SETUP
EXPORTING IDAMS RECTANGULAR FIXED FORMAT MATRICES TO FREE FORMAT MATRICES
EXPORT=(MATRIX,NAMES,CODES)  PRINT=DATA  -
   FORMAT=DELIM  WITH=SEMI  DECIM=COMMA  STRINGS=QUOTE
```

Example 4. Importing a square matrix containing distance measures for 10 objects numbered from 1 to 10; only integer values are included and are separated by the % sign; column/row codes as well as vectors of means and standard deviations are included in the matrix file.

```
$RUN IMPEX
$FILES
PRINT   = IMPMAT.LST
DATAOUT = IMPORTED.MAT                  file with the imported matrix
$SETUP
IMPORTING A FREE FORMAT MATRIX TO THE IDAMS SQUARE FIXED FORMAT MATRIX
IMPORT=(MATRIX,CODES)  MATSIZE=10  -
   FORMAT=DELIM  WITH=USER  DELCH='%'
$DATA
$PRINT
%   1%   2%   3%   4%   5%   6%   7%   8%   9%  10%
   1%
   2%38%
   3%72%25%
   4%24%53%17%
   5%64%26%76%18%
   6%48%25%63%15%61%
   7%12%50%7%42%8%8%
   8%19%7%13%4%14%1%15%
   9%29%37%34%21%24%35%3%5%
  10%32%57%29%45%26%28%74%24%61%
%46%15%7%7119%74%38%9%19%34%256%
%9%11%84%8971%23%28%12%20%35%843%
```

Chapter 17

Listing Datasets (LIST)

17.1 General Description

LIST can be used to print data values from a file, recoded variables and information from the associated IDAMS dictionary. Specific variables may be selected for printing, or the entire data and/or dictionary may be listed.

Each record in a data file is a continuous stream of data values. When printed as is, it becomes difficult to distinguish the values of adjacent variables. LIST eliminates this inconvenience by offering data printing format which separates variable values.

An IDAMS dictionary can be printed without the corresponding Data file by supplying a dummy file (i.e. an empty or null file), when defining the Data file.

17.2 Standard IDAMS Features

Case and variable selection. Cases may be selected by using a filter, or the skip cases option (SKIP). The skip option, if used, specifies that the first and every subsequent n-th case is to be printed. If a filter is specified, the skip option applies to those cases passing the filter. From the cases selected, the data values are listed for all the variables described in the dictionary or a subset if the parameter VARS is specified.

Transforming data. Recode statements may be used.

Treatment of missing data. Missing data values are printed as they occur, causing no special action.

17.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution. If all variables are selected for printing, then the complete dictionary is printed in sequential order.

Data. Numeric variables are printed with explicit decimal point, if any, and without leading zeros. If a value overflows the field width it is printed as a string of asterisks. Bad data replaced by default missing data codes are printed as blanks. Values for a variable are printed in a column that extends for as many pages as necessary for all cases selected for printing. Below is a block sketch of the printing format:

```
  v      v   v          v
xxx   xxxx   x   xxxxxxxx
xxx   xxxx   x   xxxxxxxx
xxx   xxxx   x   xxxxxxxx
 .      .    .       .
 .      .    .       .
```

The v headings on the columns represent variable numbers and the x's represent variable values. If the user requests printing of more variables than will fit on a line (127 characters), LIST will make a number of passes through the data, listing as many variables as it can each time. For example, if 50 variables were to be printed, LIST would read through the data, printing all the values, say, for the first 10 variables. Then the data would be read again for the printing, say of the next 12 variables, and so on. The number of variables printed on any pass over the data depends on the field width of the variables being printed and is automatically computed by LIST.

Sequence and case identification. Options exist to print a case sequence number and/or values of identification variable(s) with each case. (See parameters PRINT and IDVARS). They are printed as the first columns.

Recode variables. These are printed with 11 digits including an explicit decimal point and 2 decimal places.

17.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. If only a listing of the dictionary is required, the Data file is specified as NUL.

17.5 Setup Structure

```
$RUN LIST

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

17.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V5=100-199
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  PRINTING THE STUDY: 113A
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  VARS=(V3,V10-V25) IDVARS=V1
```

INFILE=**IN**/xxxx
: A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
: Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
: The maximum number of cases to be printed.
Default: All cases will be printed.

SKIP=n
: Every n-th case (or every n-th case passing the filter) is printed, starting with 1st case. The last case will always be printed unless the MAXCASES option forbids it.
Default: All cases (or all cases passing the filter) are printed.

VARS=(variable list)
: Print the data values for the specified variables. Variable values will be printed in the order they appear in this list.
Default: All variables in the dictionary are listed.

IDVARS=(variable list)
: The values of the variable(s) specified are printed to identify each case.

SPACE=**3**/n
: Number of spaces between columns.
The maximum value is SPACE=8.

PRINT=(CDICT/DICT, SEQNUM, **LONG**/SHORT, **SINGLE**/DOUBLE)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
SEQN	Print a case sequence number for each case printed. Note that cases are numbered after the filter is applied.
LONG	Assume 127 characters per print line.
SHOR	Assume 70 characters per print line.
SING	Single space between data lines.
DOUB	Double space between data lines.

17.7 Restriction

The sum of the field widths of variables to be printed, including case ID variables, must be less than or equal to 10,000 characters.

17.8 Examples

Example 1. Listing fifty variables including one recoded variable; all cases will be printed with their identification variables (V1, V2 and V4); dictionary will be printed but without C-records.

```
$RUN LIST
$FILES
PRINT  = LIST1.LST
DICTIN = STUDY.DIC                      input Dictionary file
DATAIN = STUDY.DAT                      input Data file
$RECODE
      R6=BRAC(V6,0-50=1,51-99=2)
$SETUP
LISTING THE VALUES OF 50 VARIABLES WITH 3 ID VARIABLES WITH EACH GROUP
IDVA=(V1,V2,V4)  VARS=(V3-V49,V59,V52,R6)  PRIN=DICT
```

Example 2. Listing a complete dictionary with C-records without listing the data.

```
$RUN LIST
$FILES
DICTIN = STUDY.DIC                      input Dictionary file
DATAIN = NUL
$SETUP
LISTING COMPLETE DICTIONARY
PRIN=CDICT
```

Example 3. Check recoding by listing values of input and recoded variables for 10 cases.

```
$RUN LIST
$FILES
DICTIN = A.DIC                          input Dictionary file
DATAIN = A.DAT                          input Data file
$RECODE
      R101=COUNT(1,V40-V49)
      IF MDATA(V9,V10) THEN R102=99 ELSE R102=V9+V10
      R103=BRAC(V16,15-24=1,25-34=2,35-54=3,ELSE=9)
$SETUP
CHECKING VALUES FOR 3 RECODED VARIABLES
MAXCASES=10  SKIP=10  SPACE=1  -
   VARS=(V40-V49,R101,V9,V10,R102,V16,R103)
```

Chapter 18

Merging Datasets (MERGE)

18.1 General Description

MERGE merges variables from cases in one IDAMS dataset with variables from a second dataset, matching the cases pair-wise on a common match variable(s). The cases in the two datasets do not have to be identical; that is, all cases present in one dataset do not have to be present in the other. The output data file consists of records containing user specified variables from each of the two input files along with a corresponding IDAMS dictionary. In order to distinguish the two input datasets, one is referred to as "dataset A", the other as "dataset B" throughout the write-up.

Combining datasets with identical collections of cases. An example of one use of the program is the combination of the data from the first and a subsequent wave of interviews with the same collection of respondents.

Combining datasets with somewhat different collections of cases. When there is more than one wave of interviews in a survey, some respondents may drop out, and some may be added. The program allows for these discrepancies between datasets and may, for example, be requested to output the records for all respondents, including those interviewed in only one wave. In this example, the variable values for the wave when a respondent was not interviewed would be output as missing data values.

Combining datasets with different levels of data. MERGE may also be used to combine two datasets, one of which contains data at a more aggregated level than the other. For example, household data can be added to individual household member records.

18.2 Standard IDAMS Features

Case and variable selection. A filter may be specified for either or both of the input datasets. The only difference in the format of the filter is that it must be preceded by an "A:" or "B:" in columns 1-2 to indicate the dataset to which the filter applies.

All or selected variables from each input dataset can be included in the output dataset. These output variables are specified in a variable list which has the usual format, except that variables are denoted by an "A" or "B" (instead of "V") to identify the input dataset in which they exist. For example, "A1, B5, A3-A45" selects variable V1, V3-V45 from dataset A and variable V5 from dataset B. See the output variables description in the "Program Control Statements" section.

Transforming data. Recode statements may not be used.

Treatment of missing data. For the options MATCH=UNION, MATCH=A, and MATCH=B, missing data codes are used as values for the output variables which are not available for a particular case. See the paragraph "Handling cases that appear in only one input dataset" in the section describing the output dataset below. The missing data codes are obtained from the dictionaries of the A and B datasets. The user specifies for each dataset whether the first or second missing data code should be used, and this for all variables from this dataset (see the parameters APAD and BPAD). If a variable does not have an appropriate

missing data code in the dictionary, then blanks are output.

Missing data are never output as the value for an output variable that is also one of the match variables, because a match variable value is always available from the one dataset that does contain the case. For example, with MATCH=UNION selected, suppose that variable A1 and B3 were used as the match variables and that only A1 was listed as an output variable (A1 and B3 would not both be listed as they presumably have the same value): then, if a case in dataset A was missing, the value for the A1 output variable would be the B3 value.

18.3 Results

Old (input) versus new (output) variable numbers. (Optional: see the parameter PRINT). A chart containing the input variable numbers and reference numbers, and the corresponding output variable numbers and reference numbers.

Output dictionary. (Optional: see the parameter PRINT).

Documentation of unmatched cases in either datasets A or B. There are several ways that unmatched cases, i.e. cases appearing in only one file, may be documented (see the parameter PRINT).

- The values of match variables may be printed:
  - whenever output variables from one of the datasets are padded with missing data,
  - whenever cases from dataset A are deleted,
  - whenever cases from dataset B are deleted.
- The values of variables A may be printed whenever a case from dataset A does not match any case from dataset B. The variables are printed in the order specified for the dataset in the output variables, followed by all the match variables which are not also output variables.
- The values of variables B may be printed whenever a case from dataset B does not match any case from dataset A. The variables are printed in the order specified for the dataset in the output variables, followed by all the match variables which are not also output variables.

Case counts. The program prints the number of cases existing in datasets A and B, the number of cases in dataset A and not in dataset B, the number of cases in dataset B and not in dataset A, and the total number of output cases written.

18.4 Output Dataset

The output is a new Data file and a corresponding IDAMS dictionary.

Each data record contains the values of the output variables for matching cases from datasets A and B. Note that a match variable is not automatically output: the user must include the match variable(s) from one of the datasets in the output variable list in order to give the output a case ID.

Handling cases that appear in only one input dataset. Four actions are possible:

1. MATCH=INTERSECTION. Cases that appear in only one input dataset are not included in the output dataset. (If data sets A and B are thought of as sets of cases, the output is the intersection of sets A and B).
2. MATCH=UNION. Any case that appears in either input dataset is included in the output dataset. Variables from the input dataset that does not contain the case are assigned missing data values in the output dataset. (The output is the union of sets A and B).
3. MATCH=A. Any case that appears in dataset A is included in the output dataset, while a case that appears only in dataset B is not included. If a case is found only in dataset A, variables from dataset B are assigned missing data values in the output dataset for that case. (The output is set A).

4. MATCH=B. The same as option 3, except that dataset B defines the cases included in the output dataset. (The output is set B).

Handling duplicate cases. When one of the two input datasets contains more than one case with the same value on the match variable(s), the dataset is said to contain duplicate cases. Normally (i.e. when the parameter DUPBFILE is not specified) the program prints a message about the occurrence of duplicates and then treats each of them as a separate case. The cases actually written to the output file depend on the MATCH option selected. The following figure shows how this works.

Merging Files with Duplicates (DUPBFILE not specified)

```
     Input                                    Output

  A     |    B     | MATCH = UNION|  MATCH = A   |   MATCH = B   | MATCH = INTER
        |          |              |              |               |
ID  N1  | ID  N2   | ID   N1   N2 | ID  N1   N2  | ID  N1   N2   | ID  N1   N2
        |          |              |              |               |
01 MARY | 01 JOHN  | 01 MARY JOHN | 01 MARY JOHN | 01 MARY JOHN  | 01 MARY JOHN
01 ANN  | 02 PETER | 01 ANN  ____ | 01 ANN  ____ | 02 JANE PETER | 02 JANE PETER
02 JANE | 03 MIKE  | 02 JANE PETER| 02 JANE PETER| 03 ____ MIKE  |
        |          | 03 ____ MIKE |              |               |
```

However duplicates can be interpreted and handled differently when one of the two datasets contains cases at a lower level of analysis than the other. For example, one dataset contains household data and the second contains data for household members. In this instance, the match variables specified from each file would be the household identification. Thus, "duplicates" would naturally occur in the "member of a household" dataset, as most households would have more than one member. By specifying the parameter DUPBFILE, the message about the occurrence of duplicates is not printed and cases are constructed for each "duplicate" case in dataset B with the variables from the matching A case copied onto each. The following figure shows an example of this procedure.

Merging Files at Different Levels (DUPBFILE specified)

```
     Input                                    Output

  A     |    B     | MATCH = UNION|  MATCH = A   |  MATCH = B    | MATCH = INTER
        |          |              |              |               |
ID  N1  | ID  N2   | ID   N1  N2  | ID N1   N2   | ID N1   N2    | ID N1   N2
        |          |              |              |               |
01 JONE | 01 MARY  | 01 JONE MARY | 01 JONE MARY | 01 JONE MARY  | 01 JONE MARY
03 SMIT | 01 JOHN  | 01 JONE JOHN | 01 JONE JOHN | 01 JONE JOHN  | 01 JONE JOHN
04 SCOT | 01 ANN   | 01 JONE ANN  | 01 JONE ANN  | 01 JONE ANN   | 01 JONE ANN
        | 02 PETE  | 02 ____ PETE | 03 SMIT MIKE | 02 ____ PETE  | 03 SMIT MIKE
        | 02 JANE  | 02 ____ JANE | 04 SCOT ____ | 02 ____ JANE  |
        | 03 MIKE  | 03 SMIT MIKE |              | 03 SMIT MIKE  |
        |          | 04 SCOT ____ |              |               |
```

Variable sequence and variable numbers. Variables are output in the order they are given in the output variable list and are always renumbered, starting at the value of the parameter VSTART. Thus, an output variable list such as "A1-A5, B6, A7-A25, B100" would create a dataset with variables V1 through V26 if VSTART=1. Reference numbers for variables, if they exist, are transferred unchanged to the output dictionary.

Variable locations. Variable locations are assigned by MERGE starting with the first output variable and continuing in order through the output variable list.

18.5 Input Datasets

MERGE requires 2 input Data files each described by an IDAMS dictionary.

The match variables may be alphabetic or numeric. Corresponding match variables from the A and B datasets must have the same field width.

The output variables may be alphabetic or numeric.

Each input Data file must be sorted in ascending order on its match variables prior to using MERGE.

18.6 Setup Structure

```
$RUN MERGE

$FILES
     File specifications

$SETUP
     1. Filter(s) (optional)
     2. Label
     3. Parameters
     4. Match variable specification
     5. Output variables

$DICT (conditional)
     Dictionary (see Note below)

$DATA (conditional)
     Data (see Note below)


Files:
DICTxxxx   input dictionary for dataset A (omit if $DICT used)
DATAxxxx   input data for dataset A (omit if $DATA used)
DICTyyyy   input dictionary for dataset B (omit if $DICT used)
DATAyyyy   input data for dataset B (omit if $DATA used)
DICTzzzz   output dictionary
DATAzzzz   output data
PRINT      results (default  IDAMS.LST)
```

Note. Either the A dataset or the B dataset, but not both, may be introduced in the setup. However records following $DICT and $DATA are copied into files defined by DICTIN and DATAIN respectively. Therefore, if the A file is introduced in the setup, the A dataset will be defined by DICTIN and DATAIN and INAFILE=IN must be specified. Similarly, if the B file is introduced in the setup then INBFILE=IN must be specified.

18.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter(s)** (optional). Selects a subset of cases from dataset A and/or dataset B to be used in the execution. Note that each filter statement must be preceded by "A:" or "B:" in columns one and two to indicate the dataset to which the filter applies.

```
Example: A: INCLUDE V1=10,20,30
         B: INCLUDE V1=10,20,30
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example: MERGE OF TEACHER DATA AND STUDENT DATA
```

3. **Parameters** (mandatory). For selecting program options.

```
Example: MATCH=INTE PRINT=(A, B)
```

INAFILE=**INA**/xxxx
A 1-4 character ddname suffix for the A input Dictionary and Data files.
Default ddnames: DICTINA, DATAINA.

INBFILE=**INB**/xxxx
A 1-4 character ddname suffix for the B input Dictionary and Data files.
Default ddnames: DICTINB, DATAINB.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file A.
Default: All cases will be used.

MATCH=INTERSECTION/UNION/A/B

INTE	Output only cases appearing in both datasets A and B.
UNIO	Output cases appearing in either or both datasets A and B, padding variables with missing data when necessary.
A	Output cases appearing in the A dataset only, padding B variables with missing data when necessary.
B	Output cases appearing in the B dataset only, padding A variables with missing data when necessary.

No default.

DUPBFILE
A case in dataset A may be paired with one or more cases (i.e. duplicates) from dataset B. For each pairing, an output record will be created, depending on the MATCH parameter.
Note: The dataset with the expected duplicates must be defined as the B dataset.
Default: Duplicate cases in either dataset will be noted in the printed output and then treated as distinct cases according to the MATCH specification.

OUTFILE=**OUT**/zzzz
A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

VSTART=**1**/n
Variable number for the first variable in the output dataset.

APAD=**MD1**/MD2
When padding A variables with missing data:

MD1	Output first missing data code.
MD2	Output second missing data code.

BPAD=**MD1**/MD2
When padding B variables with missing data:

MD1	Output first missing data code.
MD2	Output second missing data code.

PRINT=(**PAD**/NOPAD, **ADELETE**/NOADELETE, **BDELETE**/NOBDELETE, VARNOS, A, B, **OUTDICT**/OUTCDICT/NOOUTDICT)

PAD	Print the values of match variables when padding any A or B variables with missing data.
ADEL	Print the values of match variables for dataset A whenever a case from dataset A is not included in the output data file.
BDEL	Print the values of match variables for dataset B whenever a case from dataset B is not included in the output data file.
VARN	Print a list of the variable numbers in the input datasets and corresponding variable numbers in the output dataset.
A	Print all output and match variable values for cases appearing only in dataset A, whether or not they are included in the output dataset.
B	Print all output and match variable values for cases appearing only in dataset B, whether or not they are included in the output dataset.
OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
NOOU	Do not print the output dictionary.

4. **Match variable specification** (mandatory). This statement defines the variables from datasets A and B that are to be compared to match cases. Note that each input data file must be sorted on its match variable(s) prior to using MERGE.

```
Example:  A1=B3, A5=B1
```

which means that for a case from dataset A to match a case from dataset B, the value of variable V1 from the dataset A must be identical to the value of variable V3 from the dataset B, and similarly for the variables V5 and V1.

General format

An=Bm, Aq=Br, ...

Rules for coding

- The field width of the two variables to be compared must be identical. The comparison is done on a character basis, not a numeric one. Thus, '0.9' is not equivalent to '009', nor is '9' equal to '09'. If the field widths are not the same, use the TRANS program to change the width of one of the variables prior to using MERGE.
- Each match variable pair is separated by a comma.
- Blanks may occur anywhere in the statement.
- To continue to another line, terminate the information at a comma and enter a dash (-) to indicate continuation.

5. **Output variables** (mandatory). This defines which variables from each input dataset are to be transferred to the output and specifies their order in the output.

```
Example:  A1, B2, A5-A10, B5, B7-B10
```

which means that the output dataset will contain variable V1 from dataset A, followed by variable V2 from dataset B, followed by variables V5 through V10 from dataset A, etc., in that order.

Rules for coding

- The rules for coding are the same as for specifying variables with the parameter VARS, except that A's and B's are used instead of V's. Each variable number from dataset A is preceded by an "A" and each variable number from dataset B is preceded by a "B".
- Duplicate variables in the list count as separate variables.

18.8 Restrictions

1. The maximum number of match variables from each dataset is 20.

2. Match variables must be of the same type and field width in each file.

3. The maximum total length of the set of match variables from each dataset is 200 characters.

18.9 Examples

Example 1. Combining records from 2 datasets with an identical set of cases; in both datasets cases are identified by variables 1 and 3; all variables are to be selected from each input dataset.

```
$RUN MERGE
$FILES
DICTOUT = AB.DIC                    output Dictionary file
DATAOUT = AB.DAT                    output Data file
DICTINA = A.DIC                     input Dictionary file for dataset A
DATAINA = A.DAT                     input Data file for dataset A
DICTINB = B.DIC                     input Dictionary file for dataset B
DATAINB = B.DAT                     input Data file for dataset B
$SETUP
COMBINING RECORDS FROM 2 DATASETS WITH AN IDENTICAL SET OF CASES
MATCH=UNION
A1=B1,A3=B3
A1-A112,B201-B401
```

Example 2. Combining datasets with somewhat different collections of cases; only cases having records in both datasets are output; cases are identified by variables 2 and 4 in the first dataset, and by variables 105 and 107 respectively in the second dataset; variables in the output dataset will be re-numbered starting from the number 201, and a listing of references is requested; only selected variables will be taken from each input dataset.

```
$RUN MERGE
$FILES
      as for Example 1
$SETUP
COMBINING RECORDS FROM 2 DATASETS WITH DIFFERENT SETS OF CASES
MATCH=INTE  VSTA=201  PRIN=VARNOS
A2=B105,A4=B107
B105,B107,A36-A42,B120,B131
```

Example 3. Combining datasets with different levels of data; cases from dataset A are combined with a subset of cases from dataset B; a case from dataset A may be paired with one or more cases from dataset B; cases in dataset A which do not match with a case in selected subset of dataset B are dropped and not listed.

```
$RUN MERGE
$FILES
      as for Example 1
$SETUP
B:  INCLUDE V18=2 AND V21=3
COMBINING 2 DATASETS WITH DIFFERENT LEVELS OF DATA
MATCH=B  DUPB
A1=B15
B15,A2,A6-A12,B20-B31,B40
```

Example 4. Household income is to be calculated from a file of household members and then merged back into individual member records; AGGREG is first used to sum the income (V6) over the individuals in the household; V3 is the variable which identifies the household; the output file from AGGREG (defined by DICTAGG and DATAAGG) will contain 2 variables, the household ID (V1) and household income (V2); this file is then used as the "A" file with MERGE to add the appropriate household income (variable A2) to each original individual's record (variables B1-B46).

```
$RUN AGGREG
$FILES
PRINT   = MERGE4.LST
DICTIN  = INDIV.DIC           input Dictionary file
DATAIN  = INDIV.DAT           input Data file
DICTAGG = AGGDIC.TMP          temporary output Dictionary file from AGGREG
DATAAGG = AGGDAT.TMP          temporary output Data file from AGGREG
DICTOUT = INDIV2.DIC          output Dictionary file from MERGE
DATAOUT = INDIV2.DAT          output Data file from MERGE
$SETUP
AGGREGATING INCOME
IDVARS=V3  AGGV=V6  STATS=SUM  OUTF=AGG
$RUN MERGE
$SETUP
MERGING HOUSEHOLD INCOME TO INDIVIDUAL RECORDS
INAFILE=AGG  INBFILE=IN  DUPB MATCH=B
A1=B3
B1-B46,A2
```

Note that once file assignments have been made under $FILES, they do not need to be repeated if they are being reused in subsequent steps.

Chapter 19

Sorting and Merging Files (SORMER)

19.1 General Description

SORMER allows the user to more conveniently execute a Sort/Merge by allowing the specification of the sort or merge control-field information in the usual IDAMS parameter format. If the data file is described by an IDAMS dictionary, then a copy of the dictionary corresponding to the sorted data can be output and the sort fields may be specified by providing the appropriate variables; if not, they are specified by their location.

Sort order. The user may specify that the data are to be sorted/merged in ascending or descending order.

19.2 Standard IDAMS Features

SORMER is a utility program and contains none of the standard IDAMS features.

19.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, for sort key variables.

Sort/Merge results. Number of records sorted/merged.

19.4 Output Dictionary

A copy of the input dictionary corresponding to the output Data file.

19.5 Output Data

Output consists of one file with the same attributes as the input file(s) with the records sorted into the requested order.

19.6 Input Dictionary

If the sort fields are being specified with variable numbers, then an IDAMS dictionary containing T-records for at minimum these variables must be input. Only dictionaries describing one record per case data are allowed.

19.7 Input Data

For **sorting**, one data file is input, containing one or more fields (or variables) whose values define the desired order.

For **merging**, input consists of 2-16 data files, each with the same record format, i.e. the same record length and fields defining the sort order in the same positions. Each file must be sorted into order by the merge control fields before merging.

19.8 Setup Structure

```
$RUN SORMER

$FILES
     File specifications

$SETUP
     1. Label
     2. Parameters

$DICT (conditional)
     Dictionary for sort/merge field variables


Files for sorting:
DICTxxxx  IDAMS dictionary for sort field variables (omit if $DICT used)
SORTIN    input data
DICTyyyy  output dictionary
SORTOUT   output data

Files for merging:
DICTxxxx  IDAMS dictionary for merge field variables (omit if $DICT used)
SORTIN01  1st data file
SORTIN02  2nd data file
   .
   .
DICTyyyy  output dictionary
SORTOUT   output data

PRINT     results (default  IDAMS.LST)
```

Note. When SORMER execution is requested more than once in one setup file, the input file definitions specified in the subsequent execution only modify but not replace the input file definitions specified previously, e.g. if SORTIN01, SORTIN02 and SORTIN03 are specified for the first execution, and SORTIN01 and SORTIN02 are specified for the second execution in the same setup, the 'new' SORTIN01 and SORTIN02 as well as the 'old' SORTIN03 will be taken for merging.

19.9 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-2 below.

1. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example: SORTING WAVE ONE
   ```

2. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  KEYVARS=(V2,V3)
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary file.
   Default ddname: DICTIN.

   OUTFILE=yyyy
   A 1-4 character ddname suffix for the output Dictionary file.
   Needs to be specified to obtain in output a copy of the input Dictionary.

   SORT/MERGE
   SORT The input data are to be sorted.
   MERG Two or more data files are to be merged.

   ORDER=**A**/D
   A Sort in ascending order on sort fields.
   D Sort in descending order.

   KEYVARS=(variable list)
   List of variables to be used as sort fields (IDAMS dictionary must be supplied).
   Note: The data file must have one record per case for this option to be selected. If more than one record per case, use KEYLOC.

   KEYLOC=(s1,e1, s2,e2, ...)
   Sn Starting location of n-th sort field.
   En Ending location of n-th sort field. Must be specified even when equal to the starting location.

   Note. No defaults. Either KEYVARS or KEYLOC (but not both) must be specified.

   PRINT=CDICT/DICT
   CDIC Print the input dictionary for the sort key variables with C-records if any.
   DICT Print the input dictionary without C-records.

19.10 Restrictions

1. A maximum of 16 files may be merged.

2. A maximum of 12 Sort/Merge control fields or variables may be specified.

3. The maximum number of records depends on the disk space available for the work files SORTWK01, 02, 03, 04, 05. These work files can be assigned to a disk other than the default drive if necessary.

19.11 Examples

Example 1. Merging three pre-sorted data files of the same format; each file is described by the same IDAMS dictionary; cases are sorted in ascending order on three variables: V1, V2 and V4.

```
$RUN SORMER
$FILES
PRINT    = SORT1.LST
DICTIN   = \SURV\DICT.DIC          input Dictionary file
SORTIN01 = DATA1.DAT               input Data file 1
SORTIN02 = DATA2.DAT               input Data file 2
SORTIN03 = DATA3.DAT               input Data file 3
DICTOUT  = \SURV\DATA123.DIC       output Dictionary file
SORTOUT  = \SURV\DATA123.DAT       output Data file
$SETUP
MERGING THREE IDAMS DATA FILES:  DATA1, DATA2  AND DATA3
MERG  KEYVARS=(V1,V2,V4) OUTF=OUT
```

Example 2. Sorting a Data file in descending order on two fields: first field is 4 characters long, starting in column 12; second field is 2 characters long, starting in column 3; a dictionary is not used.

```
$RUN SORMER
$FILES
SORTIN  = RAW.DAT              input Data file
SORTOUT = SORT.DAT             output Data file
$SETUP
SORTING DATA FILE WITHOUT USING DICTIONARY
KEYLOC=(12,15,3,4)  ORDER=D
```

Chapter 20

Subsetting Datasets (SUBSET)

20.1 General Description

SUBSET subsets a Data file and corresponding IDAMS dictionary by case and/or by variable, or copies the complete files.

Sort order check. The program has an option to check that the data cases are in ascending order, based on a list of sort order variables (see the parameter SORTVARS). Adjacent cases with duplicate identification are not considered out of order. However, there is an option to delete duplicate occurrences of any case.

20.2 Standard IDAMS Features

Case and variable selection. Case subsetting is accomplished by using a filter to select a particular set of cases from the input dataset. Variable selection is done by defining a set of input variables to be transferred to the output dataset. The variables may be output in any order, and may be transferred more than once, provided that the output variable numbers are re-numbered.

Transforming data. Recode statements may not be used.

Treatment of missing data. SUBSET makes no distinction between substantive data and missing data values; all data are treated the same.

20.3 Results

Output dictionary. (Optional: see the parameter PRINT).

Subsetting statistics. The output record length, the number of output dictionary records and the number of output data records.

Old (input) versus new (output) variable numbers. (Optional: see the parameter PRINT). A chart containing the input variable numbers and reference numbers, and the corresponding output variable numbers and reference numbers.

Notification of duplicate cases. (Conditional: if the sort order of the file is being checked, all duplicate cases are documented whether or not the parameter DUPLICATE=DELETE is specified). For each case identification which appears more than once in the data, the number of duplicates, the sequential number of the case, and the case identification are printed. In addition, the program prints the number of input data records and the number of input data records deleted.

20.4 Output Dataset

The output is an IDAMS dataset constructed from the user specified subset of cases and/or variables from the input file. When all variables are copied, i.e. when OUTVARS is not specified, the output and input data records have the same structure and the dictionary output is an exact copy of the input. Otherwise, the dictionary information for the variables in the output file is assigned as follows:

Variable sequence and variable numbers. If VSTART is specified, variables are placed as they appear in the OUTVARS list and they are numbered according to the VSTART parameter. If VSTART is not specified, the output variables have the same numbers as input variables and they are sorted in ascending order by variable number.

Variable locations. Variable locations are assigned contiguously according to the order of the variables in the OUTVARS list (if VSTART is specified) or after sorting into variable number order (if VSTART is not specified).

Variable type, width and number of decimals are the same as for input variables.

Reference numbers. As from input or modified according to REFNO parameter.

C-records. Codes and their labels are copied as they are in the input dictionary.

20.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. Numeric or alphabetic variables can be used.

20.6 Setup Structure

```
$RUN SUBSET

$FILES
     File specifications

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default  IDAMS.LST)
```

20.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V1=10,20,30 AND V2=1,5,7
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  SUBSET OF 1968 ELECTION, V1-V50
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  SORT=(V1,V2), DUPLICATE=DELETE
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   MAXCASES=n
   The maximum number of cases (after filtering) to be used from the input file.
   Default: All cases will be used.

   SORTVARS=(variable list)
   If the sort order of the file is to be checked, specify up to 20 variables which define the sort sequence in major to minor order. Duplicates are considered as being in ascending order.

   DUPLICATE=**KEEP**/DELETE
   Deletion of duplicate cases (only applicable if SORT specified).

KEEP	Output all occurrences of duplicate cases.
DELE	Output only the first occurrence of duplicate cases, and print message for duplicate(s).

   OUTVARS=(variable list)
   Supply this list only if a subset of the variables in the input dataset is to be output. If VSTART is not selected, then duplicates are not allowed. Otherwise, variables can be provided in any order and repeated as needed.
   Default: All variables are output.

   OUTFILE=**OUT**/yyyy
   A 1-4 character ddname suffix for the output Dictionary and Data file.
   Default ddnames: DICTOUT, DATAOUT.

   VSTART=n
   The variables will be numbered sequentially, starting at n, in the output dataset.
   Default: Input variable numbers are retained.

   REFNO=**OLDREF**/VARNO

OLDR	Retain the reference numbers in C- and T-records as in the input dictionary.
VARN	Update the reference number field in C- and T-records to match the output variable number.

   PRINT=(**OUTDICT**/OUTCDICT/NOOUTDICT, VARNOS)

OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
VARN	Print a list of the old and new variable numbers and reference numbers.

20.8 Restrictions

1. The maximum number of sort variables that may be defined is 20.
2. The combined field widths of the sort variables must not exceed 200 characters.

20.9 Examples

Example 1. Constructing a subset of cases for selected variables; variables will be re-numbered starting at 1 and a table giving the old and new variable numbers will be printed.

```
$RUN SUBSET
$FILES
PRINT   = SUBS1.LST
DICTIN  = ABC.DIC                       input Dictionary file
DATAIN  = ABC.DAT                       input Data file
DICTOUT = SUBS.DIC                      output Dictionary file
DATAOUT = SUBS.DAT                      output Data file
$SETUP
INCLUDE V5=2,4,5 AND V6=2301
SUBSETTING VARIABLES AND CASES
PRINT=VARNOS  VSTART=1  -
   OUTVARS=(V1-V5,V18,V43-V57,V114,V116)
```

Example 2. Using the SUBSET program to check for duplicate cases; cases are identified by variables in columns 1-3 and 7-8; there is one record per case; the output dataset is not required and is not kept.

```
$RUN SUBSET
$FILES
DATAIN  = DEMOG.DAT                     input Data file
$SETUP
CHECKING FOR DUPLICATE CASES
SORT=(V2,V4)  PRIN=NOOUTDICT
$DICT
$PRINT
   3   2   4   1   1
T   2 CASE FIRST ID VAR          1   3
T   4 CASE SECOND ID VAR         7   2
```

Chapter 21

Transforming Data (TRANS)

21.1 General Description

The TRANS program creates a new IDAMS dataset containing variables from an existing dataset and new variables defined by Recode statements. It is the way to "save" recoded variables.

TRANS has a print option and so it can also be used for testing Recode statements on a small number of cases before executing an analysis program or before saving the complete file.

21.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of the cases from the input data. Variable selection is accomplished through the parameter OUTVARS.

Transforming data. Recode statements may be used.

Treatment of missing data. Appropriate missing data codes are written to the output dictionary; these are normally copied from the input dictionary but can also be overridden or supplied for output variables through the Recode statement MDCODES. No missing data checks are made on data values except through the use of Recode statements.

21.3 Results

Output dictionary. (Optional: see the parameter PRINT).

Output data. (Optional: see the parameter PRINT). Values for all cases for each V- or R-variable are given, 10 variable values per line. For alphabetic variables, only the first 10 characters are printed.

21.4 Output Dataset

The output is an IDAMS dataset which contains only those variables (V and R) specified in the OUTVARS parameter. The dictionary information for the variables in the output file is assigned as follows:

Variable sequence and variable numbers. If VSTART is specified, variables are placed as they appear in the OUTVARS list and they are numbered according to the VSTART parameter. If VSTART is not specified, the output variables have the same numbers as in the OUTVARS list and they are sorted in ascending order by variable number.

Variable names and missing data codes. Taken from the input dictionary (V-variables only) or from Recode NAME and MDCODES statements, if any.

Variable locations. Variable locations are assigned contiguously according to the order of the variables in the OUTVARS list (if VSTART is specified) or after sorting into variable number order (if VSTART is not specified).

Variable type, width and number of decimals.

V-variables: Type, field width and number of decimals are the same as their input values.

R-variables: Type for R-variables is always numeric; width and number of decimals are assigned according to the values specified for parameters WIDTH (default 9) and DEC (default 0), or according to the values provided for individual variables on dictionary specifications.

Reference numbers and study ID. The reference number and study ID for a V-variable are the same as their input values. For R-variables, the reference number is left blank and the study ID is always REC.

C-records. C-records cannot be created for R-variables. C-records (if any) for all V-variables are copied to the output dictionary. Note that if a V-variable is recoded during the TRANS execution, the C-records that are output may no longer apply to the new version of the variable.

21.5 Input Dataset

The input is a data file described by an IDAMS dictionary. Numeric or alphabetic variables can be used.

21.6 Setup Structure

```
$RUN TRANS

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Dictionary specifications (optional)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default  IDAMS.LST)
```

21.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  EXCLUDE V19=2-3
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  CONSTRUCTING VIOLENCE INDICATORS
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  VSTART=1, WIDTH=2  OUTVARS=(V2-V5,R7)
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric input data values and "insufficient field width" output values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MAXERR=**0**/n
The maximum number of "insufficient field width" errors allowed before execution stops. These errors occur when the value of a variable is too big to fit into the field assigned, e.g. a value of 250 when WIDTH=2 has been specified. See "Data in IDAMS" chapter.

OUTFILE=**OUT**/yyyy
A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

OUTVARS=(variable list)
V- and R-variables which are to be output. The order of the variables in the list is significant only if the parameter VSTART is specified. If VSTART is not specified all V- and R-variable numbers must be unique.
No default.

VSTART=n
The variables will be numbered sequentially, starting at n, in the output dataset.
Default: Input variable numbers are retained.

WIDTH=**9**/n
The default output variable field width to be used for R-variables. This default may be overridden for specific variables with the dictionary specification WIDTH. To change the field width of a numeric V-variable, create an equivalent R-variable (see Example 1).

DEC=**0**/n
Number of decimal places to be retained for R-variables.

PRINT=(**OUTDICT**/OUTCDICT/NOOUTDICT, DATA)
- OUTD Print the output dictionary without C-records.
- OUTC Print the output dictionary with C-records if any.
- DATA Print the values of the output variables.

4. **Dictionary specifications** (optional). For any particular set of variables, the field width and number of decimals may be specified. These specifications will override the values set by the main parameters WIDTH and DEC. Note that missing data codes and variable names are assigned by the Recode statements MDCODES and NAME respectively. Warning: MDCODES statement retains only 2 decimal places for R-variables, rounding up the values accordingly.

   The coding rules are the same as for parameters. Each dictionary specification must begin on a new line.

```
Examples:  VARS=R4, WIDTH=4, DEC=1
           VARS=R8, WIDTH=2
           VARS=(R100-R109), WIDTH=1
```

VARS=(variable list)
The R-variables to which the WIDTH and DEC parameters apply.

WIDTH=n
Field width for the output variables.
Default: Value given for WIDTH parameter.

DEC=n
Number of decimal places.
Default: Value given for DEC parameter.

21.8 Restrictions

1. The maximum number of R-variables that can be output is 250.
2. The maximum number of variables that can be used in the execution (including variables used only in Recode statements) is 500.
3. The maximum number of dictionary specifications is 200.

21.9 Examples

Example 1. Selected variables from the input dataset are transferred to the output file along with the 2 new variables; variable numbers are not changed; the field width of input variable V20 is changed to 4.

```
$RUN TRANS
$FILES
PRINT   = TRANS1.LST
DICTIN  = OLD.DIC                       input Dictionary file
DATAIN  = OLD.DAT                       input Data file
DICTOUT = NEW.DIC                       output Dictionary file
DATAOUT = NEW.DAT                       output Data file
$SETUP
CONSTRUCTING TWO NEW VARIABLES
PRINT=NOOUTDICT  OUTVARS=(V1-V19,R20,V33,V45-V50,R105,R122)
VARS=R105,WIDTH=1
VARS=R122,WIDTH=3,DEC=1
VARS=R20,WIDTH=4
$RECODE
```

```
R20=V20
NAME R20'VARIABLE 20'
R105=BRAC(V5,15-25=1,<36=2,<46=3,<56=4,<66=5,<90=6,ELSE=9)
MDCODES R105(9)
NAME R105'GROUPS OF AGE'
IF MDATA(V22) THEN R122=99.9 ELSE R122=V22/3
MDCODES R122(99.9)
NAME R122'NO ARTICLES PER YEAR'
```

Example 2. This example shows the use of TRANS to check Recode statements; data values for the ID variables (V1, V2), the variables being used in the recodes and the result variables are listed for the first 30 cases; the output dataset is not required and is not defined.

```
$RUN TRANS
$FILES
PRINT  = TRANS2.LST
DICTIN = STUDY.DIC                      input Dictionary file
DATAIN = STUDY.DAT                      input Data file
$SETUP
CHECKING RECODES
WIDTH=2  PRINT=(DATA,NOOUTDICT)  MAXCASES=30  -
    OUTVARS=(V1-V2,V71-V74,V118,V12,V13,R901-R903)
$RECODE
   R901=BRAC(V118,1-16=2,17=1,18-23=3,24=1,25-35=3,36=1,37=2,ELSE=9)
   IF NOT MDATA(V12,V13) THEN R902=TRUNC(V12/V13) ELSE R902=99
   R903=COUNT(1,V71-V74)
```

Example 3. Creating a test file of data with a random 1/20 sample of data file; there is no need to save the output dictionary as it will be identical to the input.

```
$RUN TRANS
$FILES
DICTIN  = STUDY.DIC                     input Dictionary file
DATAIN  = STUDY.DAT                     input Data file
DATAOUT = TESTDATA                      output Data file
$SETUP
CREATING TEST FILE WITH ALL VARIABLES AND 1/20 SAMPLE OF CASES
PRINT=NOOUTDICT  OUTVARS=(V1-V505)
$RECODE
   IF RAND(0,20) NE 1 THEN REJECT
```

Part IV

Data Analysis Facilities

Chapter 22

Cluster Analysis (CLUSFIND)

22.1 General Description

CLUSFIND performs cluster analysis by partitioning a set of objects (cases or variables) into a set of clusters as determined by one of six algorithms: two algorithms based on partitioning around medoids, one based on fuzzy clustering and three based on hierarchical clustering.

22.2 Standard IDAMS Features

Case and variable selection. If raw data are input, the standard filter is available to select a subset of cases from the input data. The variables for analysis are specified in the parameter VARS.

Transforming data. If raw data are input, Recode statements may be used.

Weighting data. Use of weight variables is not applicable.

Treatment of missing data. If raw data are input, the MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. The cases in which missing data occur in all variables are deleted automatically. Otherwise, missing data are suppressed "by pairs". If the data are standardized, the average and the mean absolute deviation are calculated using only valid values. When calculating the distances, only those variables are considered in the sum for which valid values are present for both objects.

If a matrix is input, the MDMATRIX parameter is available to indicate which value should be used to check for invalid matrix elements.

22.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Input data after standardization. (Optional: see the parameter PRINT). Standardized values for all cases for each V- or R-variable used in analysis, preceded by the average and the mean absolute deviation for those variables.

Dissimilarity matrix. (Optional: see the parameter PRINT). The lower-left triangle of the matrix, as input or computed by the program.

PAM analysis results. For each number of clusters in turn (going from CMIN to CMAX) the following is printed:

number of representative objects (clusters) and the final average distance,
for each cluster: representative object ID, number of objects and the list of objects belonging to this cluster,

coordinates of medoids (values of analysis variables for each representative object; for input dataset only),
clustering vector (vector of numbers corresponding to the objects indicating to which cluster each object belongs) and clustering characteristics,
graphical representation of results, i.e. a plot of silhouette for each cluster (optional - see the parameter PRINT).

FANNY analysis results. For each number of clusters in turn (going from CMIN to CMAX) the following is printed:
number of clusters,
objective function value at each iteration,
for each object, its ID and the membership coefficient for each cluster,
partition coefficient of Dunn and its normalized version,
closest hard clustering, i.e. number of objects and the list of objects belonging to each cluster,
clustering vector,
graphical representation of results, i.e. a plot of silhouette for each cluster (optional - see the parameter PRINT).

CLARA analysis results. For the number of clusters tried the following is printed:
list of objects selected in the sample retained,
clustering vector,
for each cluster: representative object ID, number of objects and the list of objects belonging to this cluster,
average and maximum distances to each medoid,
graphical representation of results, i.e. a plot of silhouette for each cluster belonging to the selected sample (optional - see the parameter PRINT).

AGNES analysis results contain the following:
final ordering of objects (identified by their ID) and dissimilarities between them,
graphical representation of results, i.e. a plot of dissimilarity banner (optional - see the parameter PRINT).

DIANA analysis results contain the following:
final ordering of objects (identified by their ID) and diameters of the clusters,
graphical representation of results, i.e. a plot of dissimilarity banner (optional - see the parameter PRINT).

MONA analysis results contain the following:
trace of splits (optional - see the parameter PRINT) with, for each step, the cluster to be separated, the list of objects (identified by their ID variable values) in each of the two subsets and the variable used for the separation,
the final ordering of objects,
graphical representation of results, i.e. a separation plot with the list of objects in each cluster and the variable used for the separation (optional - see the parameter PRINT).

22.4 Input Dataset

The input dataset is a Data file described by an IDAMS dictionary. All variables used for analysis must be numeric; they may be integer or decimal valued. The case ID variable can be alphabetic. Variables used in PAM, CLARA, FANNY, AGNES or DIANA analysis should be interval scaled. Variables used in the MONA analysis should be binary (with 0 or 1 values). Note that CLUSFIND uses at most 8 characters of the variable name as provided in the dictionary.

22.5 Input Matrix

This is an IDAMS square matrix. See "Data in IDAMS" chapter. It can contain measures of similarities, dissimilarities or correlation coefficients. Note that CLUSFIND uses at most 8 characters of the object name as provided on variable identification records.

22.6 Setup Structure

```
$RUN CLUSFIND

$FILES
     File specifications

$RECODE (optional with raw data input; unavailable with matrix input)
     Recode statements

$SETUP
     1. Filter (optional; for raw data input only)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary for raw data input

$DATA (conditional)
     Data for raw data input

$MATRIX (conditional)
     Matrix for matrix input


Files:
FT09       input matrix (if $MATRIX not used and a matrix input)
DICTxxxx   input dictionary (if $DICT not used and INPUT=RAWDATA)
DATAxxxx   input data (if $DATA not used and INPUT=RAWDATA)
PRINT      results (default IDAMS.LST)
```

22.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution. Available only with raw data input.

   ```
   Example:  INCLUDE V8=5-10
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  PARTITION AROUND MEDOIDS
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  ANALYSIS=PAM   VARS=(V7-V12)  IDVAR=V1
   ```

   INPUT=**RAWDATA**/SIMILARITIES/DISSIMILARITIES/CORRELATIONS

RAWD	Input: Data file described by an IDAMS dictionary.
SIMI	Input: measures of similarities in the form of an IDAMS sqaure matrix.
DISS	Input: measures of dissimilarities in the form of an IDAMS square matrix.
CORR	Input: correlation coefficients in the form of an IDAMS square matrix.

Parameters only for raw data input

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=**100**/n
The maximum number of cases (after filtering) to be used from the input file.
Its value depends on the memory available.

n=0	No execution, only verification of parameters.
0<n<=100	Normal execution.
n>100	Only CLARA analysis allowed.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

STANDARDIZE
Standardize the variables before computing dissimilarities.

DTYPE=**EUCLIDEAN**/CITY
Type of distance to be used for computing dissimilarities.

EUCL	Euclidean distance.
CITY	City block distance.

IDVAR=variable number
Variable to be printed as case ID. Only 3 characters are used on the results. Thus, integer variables must have values smaller than 1000. Only the first three characters of an alphabetic variable are printed.
No default.

PRINT=(CDICT/DICT, STAND)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
STAN	Print the input data after standardization.

Parameters only for matrix input

DISSIMILARITIES=**ABSOLUTE**/SIGN
For INPUT=CORR, specifies how dissimilarity matrix should be computed.

ABSO	Consider absolute values of correlation coefficients as similarity measures.
SIGN	Use correlation coefficients with their signs.

MDMATRIX=n
Treat matrix elements equal to n as missing data.
Default: All values are valid.

PRINT=MATRIX
Print the input matrix.

Parameters for both types of input

VARS=(variable list)
The variables to be used in this analysis.
No default.

ANALYSIS=PAM/FANNY/CLARA/AGNES/DIANA/MONA
Specifies the type of analysis to be performed.

PAM	Partition around medoids.
FANN	Partition with fuzzy clustering.
CLAR	Partition around medoids (same as PAM), but for datasets of at least 100 cases. CLUSFIND will sample the cases and choose the best representative sample. Five samples of 40+2*CMAX cases are drawn (see CMAX parameter below). Only for raw data input.
AGNE	Agglomerative hierarchical clustering.
DIAN	Divisive hierarchical clustering.
MONA	Monothetic clustering of data consisting of binary variables. Requires at least 3 variables. Only for raw data input.

No default.

CMIN=**2**/n
For PAM and FANNY. The minimum number of clusters to try.

CMAX=n
For PAM and FANNY, the maximum number of clusters to try.
For CLARA, the exact number of clusters to try.
Default: The larger of 20 and the value specified for CMIN.

PRINT=(DISSIMILARITIES, GRAPH, TRACE, VNAMES)

DISS	Print the dissimilarity matrix.
GRAP	Print the graphical representation of the results.
TRAC	Print each step of the binary split when MONA is specified.
VNAM	For matrix input, print the first 3 or 8 characters of variable names instead of variable numbers as object identification.

22.8 Restrictions

1. The maximum number of cases which can be used in an analysis (except CLARA) is 100.
2. The minimum number of cases requested for CLARA analysis is 100.
3. The maximum number of objects in an input matrix is 100.
4. Only 3 characters of the ID variable are used on the results.

22.9 Examples

Example 1. Clustering the first 100 cases into 5 groups using 6 quantitative variables V11-V16; variable values are standardized and Euclidean distance is used in calculations; clustering is done as partitioning around medoids; printing of graphics is requested; cases are identified by variable V2.

```
$RUN CLUSFIND
$FILES
PRINT   = CLUS1.LST
DICTIN  = MY.DIC                        input Dictionary file
DATAIN  = MY.DAT                        input Data file
$SETUP
PAM ANALYSIS USING RAW DATA AS INPUT
BADD=MD1  VARS=(V11-V16)  STAND  IDVAR=V2  CMIN=5  CMAX=5  PRINT=GRAP
```

Example 2. Agglomerative hierarchical clustering of 30 towns; the input matrix contains distances between the towns and the towns are numbered from 1 to 30; printing of graphics is requested; town names are used on the results.

```
$RUN CLUSFIND
$FILES
PRINT   = CLUS2.LST
FT09    = TOWNS.MAT                     input Matrix file
$SETUP
AGNES ANALYSIS USING MATRIX OF DISTANCES AS INPUT
$COMMENT  ACTUAL DISTANCES WERE DIVIDED BY 10,000 TO BE IN THE INTERVAL 0-1
INPUT=DISS  VARS=(V1-V30)  ANAL=AGNES  PRINT=(GRAP,VNAMES)
```

Chapter 23

Configuration Analysis (CONFIG)

23.1 General Description

CONFIG performs analysis on a single spatial configuration input in the form of an IDAMS rectangular matrix (as output for example by MDSCAL). It has the capability of centering, norming, rotating, translating dimensions, computing interpoint distances and computing scalar products.

Each row of a configuration matrix provides the coordinates of one point of the configuration. Thus the number of rows equals the number of points (variables), while the number of columns equals the number of dimensions.

CONFIG can provide output which allows the user to compare more easily configurations which originally had dissimilar orientations. It can also be used to perform further analysis on a configuration. Rotation, for example, may make a configuration more easily interpreted.

23.2 Standard IDAMS Features

Case and variable selection. Selecting a subset of the cases is not applicable and a filter is not available. Nor is there an option within CONFIG to subset the input configuration. An option for selection of one matrix from a file containing multiple matrices is available within CONFIG (see the parameter DSEQ).

Transforming data. Use of Recode statements is not applicable in CONFIG.

Weighting data. Use of weight variables is not applicable.

Treatment of missing data. CONFIG does not recognize missing data in the input configuration. Ordinarily this presents no problem, as configurations are usually complete.

23.3 Results

Input matrix dictionary. (Conditional: only if the input matrix contained a dictionary. See the parameter MATRIX). Input variable dictionary records with corresponding numbers used on plots (plot labels).

Input configuration. A printed copy of the input configuration.

Centered configuration. (Optional: see the parameter PRINT). If PRINT=ALL or PRINT=CENT is specified and the input configuration is already centered, the message "Input configuration is centered" is printed.

Normalized configuration. (Optional: see the parameter PRINT). If PRINT=ALL or PRINT=NORM is specified and the input configuration is already normalized, the message "Configuration is normalized" is printed.

Solution with principal axes. (Optional: see the parameter PRINT). The rows of the matrix are the points and the columns are the principal axes. The elements in the matrix are the projections of the points on the axes.

Scalar products. (Optional: see the parameter PRINT). The lower-left half of the symmetric matrix is printed. Each element of the matrix is the scalar product for a pair of points (variables).

Inter-point distances. (Optional: see the parameter PRINT). The lower-left half of the symmetric matrix is printed. Each element in the matrix is the distance between a pair of points (variables). The diagonal, always all zeros, is printed.

Transformed configuration(s). (Optional: see the transformation specification parameter PRINT). The transformed configuration is printed after the rotation/translation.

Plot of the transformed configuration(s). (Optional: see the transformation specification parameter PRINT). The transformed configuration is plotted 2 axes at a time after the rotation/translation. The points are numbered.

Varimax rotation history. (Optional: see the parameter PRINT). A vector is printed which contains the variance of the configuration matrix before each iteration cycle. This is followed by the configuration matrix after rotation to maximize the normal varimax criterion. It will have the same number of rows and columns as the input configuration matrix.

Sorted configuration. (Optional: see the parameter PRINT). Each column of the configuration matrix, after being ordered, is printed horizontally across the page.

Vector plots. (Optional: see the parameter PRINT). The final configuration is plotted two axes at a time. The points are numbered using the plot labels for the variables as printed with the input configuration dictionary.

23.4 Output Configuration Matrix

The final configuration may be written to a file (see the parameter WRITE). It is output as an IDAMS rectangular matrix. See "Data in IDAMS" chapter for a description of IDAMS matrices. Variable identification records are output only if such records are included in the input configuration file (see the parameter MATRIX). The format for the matrix elements is 10F7.3. The records containing the matrix elements are identified by CFG in columns 73-75 and a sequence number in columns 76-80. The dimensions of the matrix will be the same as the dimensions of the input matrix.

23.5 Output Distance Matrix

The inter-point distance matrix may be written to a file (see the parameter WRITE). This is output in the form of an IDAMS square matrix with dummy records supplied for the means and standard deviations expected in such a matrix. Variable identification records are output only if these are included in the input configuration file (see the parameter MATRIX). The format of the matrix elements is 10F7.3. The records containing the matrix elements are identified by CFG in columns 73-75 and a sequence number in columns 76-80.

23.6 Input Configuration Matrix

The input matrix must be in the form of an IDAMS rectangular matrix, either with or without variable identification records (see the parameter MATRIX). See "Data in IDAMS" chapter for a description of the format.

Configuration matrices obtained from the MDSCAL program can be input directly to CONFIG.

The n(rows) by m(columns) input matrix should contain the coordinates of n points for m dimensions. There may be no missing data in the input matrix.

More than one configuration can exist in a file being input to CONFIG. The one to be analyzed is selected using the parameter DSEQ.

23.7 Setup Structure

```
$RUN CONFIG

$FILES
     File specifications

$SETUP
     1. Label
     2. Parameters
     3. Transformation specifications (conditional)

$MATRIX (conditional)
     Matrix


Files:
FT02       output configuration and/or distance matrix
FT09       input configuration (omit if $MATRIX used)
PRINT      results (default  IDAMS.LST)
```

23.8 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  CONFIG EXECUTED AFTER MDSCAL
   ```

2. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  PRINT=(CENT,SORT,DIST) TRANS
   ```

   MATRIX=**STANDARD**/NONSTANDARD
   - STAN Variable identification records are included in the input configuration matrix.
   - NONS Variable identification records are not included.

   DSEQ=**1**/n
   The sequence number on the input file of the configuration which is to be analyzed.

   WRITE=(CONFIG,DISTANCES)
   - CONF Output the final configuration to a file.
   - DIST Output the matrix of inter-point distances to a file.

   TRANSFORM
   Transformation specifications will be provided.

PRINT=(CENTER, NORMALIZE, PRINAXIS, SCALARS, DISTANCES, VARIMAX, SORTED, PLOT, ALL)

CENT	Shift origin to centroid of space.
NORM	Alter size of the space so sum of squared elements of the matrix equals the number of variables.
PRIN	Look for principal axes.
SCAL	Matrix of scalar products.
DIST	Matrix of inter-point distances.
VARI	Orthogonal (varimax) rotation (after transformation if any).
SORT	Sorted configuration (after transformation if any).
PLOT	Plot the final configuration.
ALL	Print CENT, NORM, PRIN, SCAL, DIST, VARI, SORT, PLOT.

Default: Input configuration is printed.

Note. Analysis options are performed on the input configuration in the sequence specified above, regardless of the order in which they are specified with the PRINT parameter. Transformations, if any, are performed just before orthogonal rotation of the configuration. After each operation, the results are printed. The effects of the analysis options are cumulative. If the final configuration is plotted and/or saved, this is done after all the analyses have been performed.

3. **Transformation specifications.** (Conditional: if TRANSFORM was specified, use parameters as specified below). As many transformations as desired may be specified; each one must start on a new line.

   If the user specifies the angle of rotation (DEGREES) and two dimensions (DIMENSION), rotation is performed. If a constant (ADD) and one dimension (DIMENSION) are specified, translation is performed.

```
Example:  DEGR=45, DIME=(5,8) PRINT=PLOT
```

   PRINT=(CONFIG, PLOT)

CONF	Print the translated or rotated configuration (automatic for configurations with 2 dimensions and for the final configuration).
PLOT	Plot the translated or rotated configuration.

   Note: There will be no printed output for the transformation if PRINT is not specified. It must be specified for each transformation.

   Rotation parameters

   DIMENSION=(n, m)
   The two dimensions to be rotated (only pairwise rotation).

   DEGREES=n
   Angle of rotation in degrees (only orthogonal rotation).

   Translation parameters

   DIMENSION=n
   The one dimension to be translated.

   ADD=n
   Value to be added to each coordinate for the specified dimension (may be negative and have decimal places).

23.9 Restrictions

The maximum size of the input configuration matrix is 60 rows x 10 columns.

23.10 Examples

Example 1. Rotation and transformation of a configuration matrix previously created by the MDSCAL program; the final configuration is written into a file and plotted; dimensions 1 and 2 are to be rotated by 60 degrees; dimension 1 is to be transformed by adding 6.

```
$RUN CONFIG
$FILES
PRINT  = CONF1.LST
FT02   = CONFIG.MAT         output file for configuration matrix
FT09   = MDS.MAT            input configuration matrix
$SETUP
CONFIGURATION ANALYSIS
PRINT=(PLOT,VARI)  TRAN  WRITE=CONF
DEGR=60  DIME=(1,2)  PRINT=PLOT
ADD=6 DIME=1 PRINT=PLOT
```

Example 2. Computation of the matrix of scalar products and the matrix of inter-point distances for the 4th configuration from the input file; no plots are requested.

```
$RUN CONFIG
$FILES
PRINT  = CONF2.LST
FT02   = SCAL.MAT           output file for scalar products and distances
FT09   = MDS.MAT            input configuration matrix
$SETUP
CONFIGURATION ANALYSIS
PRINT=(SCAL,DIST)  DSEQ=4
```

Chapter 24

Discriminant Analysis (DISCRAN)

24.1 General Description

The task of discriminant analysis is to find the best linear discriminant function(s) of a set of variables which reproduce(s), as far as it is possible, an a priori grouping of the cases considered.

A stepwise procedure is used in this program, i.e. in each step the most powerful variable is entered into the discriminant function. The criterion function for selecting the next variable depends on the number of groups specified (number of groups varies between 2 and 20). In the case of two groups the Mahalanobis distance is used. When the number of groups is greater than 2 then the variable selection criterion is the trace of a product of the covariance matrix for the variables involved and the inter-class covariance matrix at a particular step. This is a generalization of Mahalanobis distance defined for two groups.

Besides executing the main discriminant analysis steps on a **basic sample** there are two optional possibilities: checking the power of the discriminant function(s) with the help of a **test sample**, in which the group assignment of the cases is known (as in the basic sample) but which cases were not used in the analysis, and classifying the cases with the help of discriminant function(s) provided by the analysis in an **anonymous sample** where the group assignment of the cases is unknown, or at least is not used.

24.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. A further subsetting is possible with the use of the sample and group variables. Analysis variables are selected with the VARS parameter.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases with missing data in the sample variable, the group variable and/or the analysis variables can be optionally excluded from the analysis.

24.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Number of cases in samples. The number of cases in the basic, test and anonymous samples according to the sample definition parameters.

Revised number of cases in samples. The number of cases in the basic, test and anonymous samples revised according to the sample and group definition parameters. Note that the revised figures may be smaller than the non-revised ones for the basic and the test samples if the groups defined do not cover completely the samples.

Basic sample. (Optional: see the parameter PRINT). The identification and the analysis variables of the cases in the basic sample are printed by groups, while the groups are separated from each other by a line of asterisks.

Test sample. As for basic sample.

Anonymous sample. As for basic sample except that there are no groups.

Univariate statistics. For each variable used in the analysis the program prints the group means and standard deviations as well as the total mean.

Stepwise procedure results (for each step)

Step number. The sequence number of the step.

Variables entered. The list of variables retained in this step.

Linear discriminant function. (Conditional: only if 2 groups specified). The constant term and the coefficients of the linear discriminant function corresponding to the variables already entered.

Classification table for basic sample. Bivariate frequency table showing the re-distribution of cases between the original groups and the groups to which they are allocated on the basis of the discriminant function, followed by the percentage of the correctly classified cases.

Classification table for test sample. As for basic sample.

Case assignment list. (Optional: see the parameter PRINT). The cases of the three samples are printed here with case identification, case allocation, and discriminant function value (for 2 groups) or distances to each group (for more than 2 groups).

Discriminant factor analysis results. (Conditional: only if more than 2 groups specified). Overall discriminant power and the discriminant power of the first three factors, followed by the values of discriminant factors for group means. In addition, a graphical representation of cases and means in the space of the first two factors is also given.

24.4 Output Dataset

A dataset with the final assignment of groups to cases can be requested. It is output in the form of a data file described by an IDAMS dictionary (see parameter WRITE and "Data in IDAMS" chapter).

It contains in the following order:

- the transferred variables,
- the code of the original groups as renumbered by DISCRAN ("Original group"),
- the code of groups assigned to cases at the end ("Assigned group"),
- the "Sample type" (1=basic, 2=test, 3=anonymous) and,
- for analysis with more than 2 original groups, the values of the first two discriminant factors ("Factor-1", "Factor-2").

The variables are renumbered starting from one.

The code of the original groups is set to the first missing data code (999.9999) for cases in anonymous sample; factors are set to the first missing data code (999.9999) for cases in the test and anonymous samples.

Note: variable specified in IDVAR is not output automatically and thus ID variables should better be included in the transfer variable list.

24.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. Three types of sample can be specified in the input file, namely:

- basic sample,
- test sample, and
- anonymous sample.

The analysis is based on the basic sample. The test sample is used for testing the discriminant function(s) while the cases of the anonymous sample are simply classified using the discriminant functions.

The samples are defined by a "sample variable". The basic sample must not be empty. The groups to be separated by the discriminant function(s) should be defined by a "group variable". This variable defines an a priori classification of the basic and test sample cases.

All variables used for analysis must be numeric; they may be integer or decimal valued. The case ID variable and variables to be transferred can be alphabetic.

24.6 Setup Structure

```
$RUN DISCRAN

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary if WRITE=DATA specified
DATAyyyy   output data if WRITE=DATA specified
PRINT      results (default  IDAMS.LST)
```

24.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  INCLUDE V3=6 OR V11=99
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  DISCRIMINANT ANALYSIS ON AGRICULTURAL SURVEY
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  MDHA=SAMPVAR  IDVAR=V4  SAVAR=R5  BASA=(1,5) VARS=(V12-V15)
```

INFILE=**IN**/xxxx
: A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
: Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
: The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

VARS=(variable list)
: List of V- and/or R-variables to be used in the analysis.
No default.

MDVALUES=**BOTH**/MD1/MD2/NONE
: Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=(SAMPVAR, GROUPVAR, ANALVARS)
: Choice of missing data treatment.
SAMP Cases with missing data in the sample variable are excluded from the analysis.
GROU Cases of basic and test samples with missing data in the group variable are excluded from the analysis.
ANAL Cases with missing data in the analysis variables are excluded from the analysis.
Default: Cases with missing data are included.

WEIGHT=variable number
: The weight variable number if the data are to be weighted.

IDVAR=variable number
: Case identification variable for the data and/or case assignment listing.
Default: "DISC" is used as identifier for all cases.

STEPMAX=n
: Maximum number of steps to be performed. It must be less than or equal to the number of analysis variables.
Default: Number of analysis variables.

MEMORY=**20000**/n
: Memory necessary for program execution.

WRITE=DATA
: Create an IDAMS dataset containing transferred variables, case assignment variables, sample type and values of the discriminant factors, if any.

OUTFILE=**OUT**/yyyy
: A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

TRANSVARS=(variable list)
: Variables (up to 99) to be transferred to the output dataset.

PRINT=(CDICT/DICT, OUTCDICT/OUTDICT, DATA, GROUP)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
OUTD	Print the output dictionary without C-records.
DATA	Print the data with original group assignments of cases.
GROU	Print for each case the group assignment based on discriminant function.

Sample specification

These parameters are optional. If they are not specified, all cases from the input file are taken for the basic sample. Test and anonymous samples, if they exist, must always be explicitly defined. The pair-wise intersection of the samples must be empty. However, they need not cover the whole input data file. A single value or a range of values can be used for selecting the cases which belong to the corresponding sample.

$$\text{m1} = \text{value of sample variable}$$
$$\text{or}$$
$$\text{m1} <= \text{value of sample variable} < \text{m2}$$

where m1 and m2 may be integer or decimal values.

SAVAR=variable number
: The variable used for sample definition. V- or R-variable can be used.

BASA=(m1, m2)
: Conditional: defines the basic sample. Must be provided if SAVAR specified.

TESA=(m1, m2)
: Conditional and optional: if SAVAR is specified. Defines the test sample.

ANSA=(m1, m2)
: Conditional and optional: if SAVAR is specified. Defines the anonymous sample.

Basic sample classification

These parameters define the a priori groups used in the discriminant analysis procedure. All the groups must be defined explicitly and their pair-wise intersection must be empty. However, they need not cover the whole basic sample.

GRVAR=variable number
: The variable used for group definition. V- or R-variable can be used.
No default.

GR01=(m1, m2)
: Defines the first group in the basic sample.

GR02=(m1, m2)
Defines the second group in the basic sample.

GRnn=(m1, m2)
Defines the n-th group in the basic sample (nn <= 20).

Note. At least two groups have to be specified.

24.8 Restrictions

1. Maximum number of a priori groups is 20.
2. Same variable cannot be used twice.
3. Maximum field width of case ID variable is 4.
4. Maximum number of variables to be transferred is 99.
5. R-variables cannot be transferred.
6. If a variable to be transferred is alphabetic with width > 4, only the first four characters are used.

24.9 Examples

Example 1. Discriminant analysis on all cases together; cases are identified by the V1; 5 steps of analysis are requested; a priori groups are defined by the variable V111 which includes categories 1-6.

```
$RUN DISCRAN
$FILES
PRINT  = DISC1.LST
DICTIN = MY.DIC                  input Dictionary file
DATAIN = MY.DAT                  input Data file
$SETUP
CANONICAL LINEAR DISCRIMINANT ANALYSIS
PRINT=(DATA,GROUP)  IDVAR=V1  STEP=5  VARS=(V101-V105)  -
   GVAR=V111  GR01=(1,3)  GR02=(3,5)  GR03=(5,7)
```

Example 2. Repeat analysis described in the Example 1 using the subset of respondents having the value 1 on V5 as the basic sample and test the results on the respondents having the value 2 on V5.

```
$RUN DISCRAN
$FILES
     as for Example 1
$SETUP
CANONICAL LINEAR DISCRIMINANT ANALYSIS USING BASIC AND TEST SAMPLES
PRINT=(DATA,GROUP)  IDVAR=V1  STEP=5  VARS=(V101-V105)  -
   SAVAR=V5  BASA=1  TESA=2  -
   GVAR=V111  GR01=(1,3)  GR02=(3,5)  GR03=(5,7)
```

Chapter 25

Distribution and Lorenz Functions (QUANTILE)

25.1 General Description

QUANTILE generates distribution functions, Lorenz functions, and Gini coefficients for individual variables, and performs the Kolmogorov-Smirnov test between two variables or between two samples.

25.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. In addition, each analysis may be performed on a further subset by use of a filter parameter. Variables to be analysed are specified with VAR parameter.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer values not grater than 32,767. Note that decimal valued weights are rounded to the nearest integer. When the value of the weight variable for a case is zero, negative, missing, non-numeric or exceeding the maximum, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases containing a missing data value on analysis variable are eliminated from that analysis.

25.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Results for each analysis.

Distribution function: minimum, maximum, and subinterval break points.
Lorenz function (optional): minimum, maximum, subinterval break points, and Gini coefficient.
Lorenz curve (optional): plotted in deciles.
Kolmogorov-Smirnov test statistics (optional).

25.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. All variables referenced (except main filter) must be numeric; they may be integer or decimal valued.

25.5 Setup Structure

```
$RUN QUANTILE

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Subset specifications (optional)
     5. QUANTILE
     6. Analysis specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

25.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 and 6 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V5=1
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  MAKING DECILES
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  MDVAL=MD1, PRINT=DICT
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter. Cases with missing data in an analysis are eliminated from that analysis.

PRINT=CDICT/DICT
CDIC Print the input dictionary for the variables accessed with C-records if any.
DICT Print the input dictionary without C-records.

4. **Subset specifications** (optional). These statements permit selection of a subset of cases for a particular analysis.

```
Example:  FEMALE   INCLUDE V6=2
```

Rules for coding

Prototype: name statement

name
Subset name. 1-8 alphanumeric characters beginning with a letter. This name must match exactly the name used on subsequent analysis specifications. Embedded blanks are not allowed. It is recommended that all names be left-justified.

statement
Subset definition which follows the syntax of the standard IDAMS filter statement.

5. **QUANTILE.** The word QUANTILE on this line signals that analysis specifications follow. It must be included (in order to separate subset specifications from analysis specifications) and must appear only once.

6. **Analysis specifications.** The coding rules are the same as for parameters. Each analysis specification must begin on a new line.

```
Examples:  VAR=R10  N=5   PRINT=CLORENZ
           VAR=V25  N=10  FILTER=MALE       ANALID=M
           VAR=V25  N=10  FILTER=FEMALE  KS=M
```

VAR=variable number
Variable to be analysed.
No default.

WEIGHT=variable number
The weight variable number if the data are to be weighted. Data weighting is not allowed for the Kolmogorov-Smirnov test.

N=**20**/n
Number of subintervals. If n<2 or n>100, a warning is printed and the default value of 20 is used.

FILTER=xxxxxxxx
: Only cases which satisfy the condition defined on the subset specification named xxxxxxxx will be used for this analysis. Enclose the name in primes if it contains non-alphanumeric characters. Upper case letters should be used in order to match the name on the subset specification which is automatically converted to upper case.

ANALID='label'
: A label for this analysis so that it can be referenced for doing a Kolmogorov-Smirnov test. Must be enclosed in primes if it contains non-alphanumeric characters.

KS='label'
: Label is the label assigned to a previous analysis through the ANALID parameter and defines the variable and/or sample with which this analysis is to be compared using the Kolmogorov-Smirnov test. Must be enclosed in primes if it contains non-alphanumeric characters.

PRINT=(FLORENZ, CLORENZ)

FLOR	Print the Lorenz function and Gini coefficient.
CLOR	Print the Lorenz curve plotted in deciles. (Lorenz function is also printed).

Note: If KS is specified, the PRINT parameter is ignored.

25.7 Restrictions

1. Maximum number of variables used (analysis+weight+local filter) is 50.
2. Maximum number of cases that can be analyzed is 5000.
3. Minimum number of subintervals is 2; maximum is 100.
4. Maximum number of subset specifications is 25.
5. If using the Kolmogorov-Smirnov test, the maximum number of cases that can be analyzed is 2500.
6. The Lorenz function and the Kolmogorov-Smirnov test cannot be requested for the same analysis.
7. The break point values are always printed with three decimal places. Variables with more than three decimals are truncated to three places when printed.

25.8 Example

Generation of distribution function, Lorenz function and Gini coefficients for variable V67; separate analyses are performed on all the data and then on two subsets; the Kolmogorov-Smirnov test is performed to test the difference of distributions of variable V67 in the two subsets of data.

```
$RUN QUANTILE
$FILES
PRINT  = QUANT.LST
DICTIN = MY.DIC                 input Dictionary file
DATAIN = MY.DAT                 input Data file
$SETUP
COMPARISON OF AGE DISTRIBUTIONS FOR FEMALE AND MALE
*                          (default values taken for all parameters)
FEMALE   INCLUDE V12=1
MALE     INCLUDE V12=2
QUANTILE
VAR=V67  N=15  PRINT=(FLOR,CLOR)
VAR=V67  N=15  PRINT=(FLOR,CLOR)  FILT=FEMALE  ANALID=F
VAR=V67  N=15  PRINT=(FLOR,CLOR)  FILT=MALE
VAR=V67  N=15                     FILT=MALE    KS=F
```

Chapter 26

Factor Analysis (FACTOR)

26.1 General Description

FACTOR covers a set of principal component factor analyses and analysis of correspondences having common specifications. It provides the possibility of performing, with only one read of the data factor analysis of correspondences, scalar products, normed scalar products, covariances and correlations.

For each analysis the program constructs a matrix representing the relations among the variables and computes its eigenvalues and eigenvectors. It then calculates the "case" and "variable" factors giving for each "case" and "variable" its ordinate, its quality of representation and its contribution to the factors. A graphic representation of the factors with ordinary or simplicio-factorial options can also be printed.

The **principal variables/cases** are the variables/cases on the basis of which the factorial decomposition procedure is performed, i.e. they are used in computing the matrix of relations. One can also look for a representation of other variables/cases in the factor space corresponding to the principal variables. Such variables/cases (having no influence on the factors) are called **supplementary variables/cases**.

One speaks about **ordinary representation** (of variables/cases) if the values (factor scores) coming directly from the analysis are used in the graphic representation. However, for a better understanding of the relation between variables and cases, another simultaneous representation, the **simplicio-factorial representation**, is possible.

26.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. Variables are selected with the PVARS and SVARS parameters.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. There are two ways of handling missing data:

- cases with missing data in principal variables are excluded from the analysis,
- cases with missing data in principal and/or supplementary variables are excluded from the analysis.

26.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Summary statistics. (Optional: see the parameter PRINT). Variable number, variable label, new variable number (re-numbered from 1), minimum and maximum values, mean, standard deviation, coefficient of variability, total, variance, skewness, kurtosis and weighted number of valid cases for each variable. Note that standard deviation and variance are estimates based upon weighted data.

Input data. (Optional: see the parameter PRINT). Groups of 16 variables with, on each row: the corresponding number of cases, the total for principal variables and the values of all the variables, preceded by the total for the columns (calculated for only the principal cases). Values are printed with explicit decimal point and with one decimal place. If more than 7 characters are required for printing a value, it is replaced by asterisks.

Matrix of relations (core matrix). (Optional: see the parameter PRINT). The matrix (after multiplication by ten to the n-th power as indicated in the line printed before the matrix), the trace value and the table of eigenvalues and eigenvectors.

Histogram of eigenvalues. The histogram with the percentages and cumulative percentages of each eigenvalue's contribution to the total inertia. The dashes in the histogram show the Kaiser criteria for the correlation analysis.

Dictionaries of the output data files. (Optional: see the parameter PRINT). The dictionary pertaining to the "case" factors followed by that of the "variable" factors.

Table(s) of factors. Depending upon the option(s) chosen, there will be: one table (either for "case" factors or for "variable" factors), or two tables (for both "case" and "variable" factors, in that order). According to the printing option chosen, these tables will contain only the principal cases (variables), only the supplementary ones, or both.

Table of "case" factors. It gives, line by line:

case ID value,
information relevant to all factors taken together, i.e. the quality of representation of the case in the space defined by the factors, the weight of the case and the "inertia" of the case,
information for each factor in turn, i.e. the ordinate of the case, the square cosine of the angle between the case and the factor, and the contribution of the case to the factor.

Table of "variable" factors. It gives, line by line, similar information for the variables.

Scatter plots. (Optional: see the parameter PLOTS). The first line gives the number of the factor represented along the horizontal axis with its eigenvalue and its min-max range. The second line gives the same information concerning the vertical axis. Along with the label of the execution, the number of cases/variables (i.e. points) that are represented is given. At the right side of each graph are printed:

number of points which cannot be printed for that ordinate (overlapping points),
number of points which it was not possible to represent,
page number.

Rotated factors. (Optional: see the parameter ROTATION). The variance calculated for each factor matrix in each iteration of the rotation (using the VARIMAX method) is printed, followed by the communalities of the variables before and after rotation, ending with the table of rotated factors.

Termination message. At the end of each analysis a termination message is printed with the type of analysis performed.

26.4 Output Dataset(s)

Two Data files, each with an associated IDAMS dictionary can optionally be constructed. In the "case" factors dataset, the records correspond to the cases (both principal and supplementary), the columns correspond to variables (including the case identification and transferred variables) and factors. In the "variable" factors dataset, the records correspond to the analysis variables, while the columns contain the variable

identifications (original variable numbers) and factors.

Output variables are numbered sequentially starting from 1 and they have the following characteristics:

- *Case identification (ID) and transferred variables:* V-variables have the same characteristics as their input equivalents, Recode variables are output with WIDTH=9 and DEC=2.
- *Computed factor variables:*

Name	specified by FNAME
Field width	7
No. of decimals	5
MD1 and MD2	9999999

26.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. All variables used for analysis must be numeric; they may be integer or decimal valued. They should be dichotomous or measured on an interval scale. The case ID variable and variables to be transferred can be alphabetic. There are two kinds of analysis variables, namely, principal and supplementary. In addition one variable identifying the case must exist. Other variables can be selected for transfer to the output data file of "case" factors. One or more cases at the end of the input data file can be specified as supplementary cases.

For analysis of correspondence, two types of data are suitable: a) dichotomous variables from a raw data file or b) a contingency table described by a dictionary and input as an IDAMS dataset.

26.6 Setup Structure

```
$RUN FACTOR

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. User-defined plot specifications (conditional)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary for case factors
DATAyyyy   output data for case factors
DICTzzzz   output dictionary for variable factors
DATAzzzz   output data for variable factors
PRINT      results (default  IDAMS.LST)
```

26.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  EXCLUDE V10=99 OR V11=99
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  AGRICULTURAL SURVEY 1984
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  ANAL=(CRSP,SSPRO) TRANS=(V16,V20) IDVAR=V1  -
               PVARS=(V31-V35)
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=**PRINCIPAL**/ALL

PRIN	Cases with missing data in the principal variables are excluded from the analysis while cases with missing data in supplementary variables are included. Supplementary variables factors are based on valid data only.
ALL	All cases with missing data are excluded.

ANALYSIS=(**CRSP**/NOCRSP, SSPRO, NSSPRO, COVA, CORR)
Choice of analyses.

CRSP	Factor analysis of correspondences.
SSPR	Factor analysis of scalar products.
NSSP	Factor analysis of normed scalar products.
COVA	Factor analysis of covariances.
CORR	Factor analysis of correlations.

PVARS=(variable list)
List of V- and/or R-variables to be used as the principal variables.
No default.

SVARS=(variable list)
List of V- and/or R-variables to be used as supplementary variables.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

NSCASES=**0**/n

Number of supplementary cases. Note: These cases are not included in the computations of statistics, matrix and factors; they are the last "n" ones in the data file.

IDVAR=variable number

Case identification variable for points on the plots and for cases in the output file.
No default.

KAISER/NFACT=n/VMIN=n

Criterion for determining the number of factors.

KAIS	Kaiser's criterion - number of roots greater than 1.
NFAC	Number of factors desired.
VMIN	The minimum percentage of variance to be explained by the factors taken all together. Do not type the decimal, e.g. "VMIN=95".

ROTATION=**KAISER**/UDEF/NOROTATION

Specifies VARIMAX rotation of "variable" factors. Only for correlation analysis.

KAIS	Number of factors to be rotated is defined according to the KAISER criteria.
UDEF	Number of factors to be rotated is specified by the user (see the parameter NROT).

NROT=**1**/n

Number of factors to be rotated (if ROTATION=UDEF specified).

WRITE=(OBSERV, VARS)

Controls output of files of "case" and "variable" factors. If more than one analysis is requested on the ANALYSIS parameter, these files will only be for the first specified.

OBSE	Create a file containing "case" factors.
VARS	Create a file containing "variable" factors.

OUTFILE=**OUT**/yyyy

A 1-4 character ddname suffix for the Dictionary and Data files for "case" factors.
Default ddnames: DICTOUT, DATAOUT.

OUTVFILE=**OUTV**/zzzz

A 1-4 character ddname suffix for the Dictionary and Data files for "variable" factors.
Default ddnames: DICTOUTV, DATAOUTV.

TRANSVARS=(variable list)

Variables (up to 99) to be transferred to the output "case" factor file.

FNAME=uuuu

A 1-4 character string used as a prefix for variable names of factors in output dictionaries. Must be enclosed in primes if it contains any non-alphanumeric characters. Factors have names uuuuFACT0001, uuuuFACT0002, etc.
Default: Blank.

PLOTS=**STANDARD**/USER/NOPLOTS

Controls graphical representation of results.

STAN	Standard plots will be printed for factor pairs 1-2, 1-3, 2-3 with options PAGES=1, OVLP=LIST, NCHAR=4, REPR=COORD, VARPLOT=(PRINCIPAL,SUPPL).
USER	User-defined plots are desired (see parameters for user-defined plots below).

PRINT=(CDICT/DICT, OUTCDICTS/OUTDICTS, STATS, DATA, MATRIX, **VFPRINC**/NOVFPRINC, VFSUPPL, OFPRINC, OFSUPPL)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
OUTC	Print output dictionaries with C-records if any.
OUTD	Print output dictionaries without C-records.
STAT	Print statistics of principal and supplementary variables.
DATA	Print input data.
MATR	Print the matrix of relations (core matrix) and eigenvectors.
VFPR	Print "variable" factors for the principal variables.
VFSU	Print "variable" factors for supplementary variables.
OFPR	Print "case" factors for the principal cases.
OFSU	Print "case" factors for supplementary cases.

4. **User-defined plot specifications** (conditional: if PLOT=USER specified as parameter). Repeat for each two-dimensional plot to be printed. The coding rules are the same as for parameters. Each plot specification must begin on a new line.

```
Example:  X=3   Y=10
```

X=factor number

Number of the factor to be represented on the horizontal axis.

Y=factor number

Number of the factor to be represented on the vertical axis (see also the plot parameter FORMAT=STANDARD).

ANSP=**ALL**/CRSP/SSPRO/NSSPRO/COVA/CORR

Specifies the analyses for which the plots are to be printed.

ALL Plots for all analyses specified in the ANALYSIS parameter.

For the rest, a plot for a single analysis (keywords have same meaning as for ANALYSIS parameter). These options imply one plot only.

OBSPLOT=(PRINCIPAL, SUPPL)

Choice of cases to be represented on the plot(s).

PRIN	Represent principal cases.
SUPP	Represent supplementary cases.

VARPLOT=(**PRINCIPAL**/NOPRINCIPAL, SUPPL)

Choice of variables to be represented on the plot(s).

PRIN	Represent principal variables.
SUPP	Represent supplementary variables.

REPRESENT=**COORD**/BASVEC/NORMBV

Choice of simultaneous representation of points (variables/cases).

COOR	Coordinates as indicated in the table of factors.
BASV	Represent basic vectors.
NORM	Represent basic vectors using special norm for "simplicio-factorial" representation.

OVLP=**FIRST**/LIST/DEN

Option concerning the representation of overlapping points.

FIRS	Print the variable number/case ID of the first point only.
LIST	Give a vertical list of the points having the same abscissa in the graph until another point is met (the variable number/case ID's are then lost).
DEN	Print the density (number of overlapping points). Print for one point ".", for two (overlapping) points ":", for three points "3", etc, for 9 points "9", for more than 9 points "*". NCHAR=2 must be specified if this option is selected.

NCHAR=**4**/n
Number of digits/characters used for the identification of the variables/cases on the plot(s) (1 to 4 characters).

PAGES=**1**/n
Number of pages per plot.

FORMAT=**STANDARD**/NONSTANDARD
Defines frame size of the plot.

STAN	Use a 21 x 30 cm frame for the plot showing the factor with the wider range on the horizontal axis and using different scales for the two axes.
NONS	The frame will not be standardized in the sense above. Size of plot is defined by PAGES=n, and meaning of axes by X and Y.

26.8 Restrictions

1. Maximum number of analysis variables is 80.
2. One (and only one) identification variable must be specified.
3. Maximum number of variables to be transferred is 99.
4. Maximum number of input variables including those used in filter and Recode statements is 100.
5. Maximum of 24 user-defined plots.
6. If the ID variable or a variable to be transferred is alphabetic with width > 4, only the first four characters are used.
7. For the parameters the following must hold:

   $\max(D1,D2,D3) < 5000$

   where
   $D1 = NPV * NPV + 10 * NV$
   $D2 = NV * (NF + 6) + NPV * NIF$
   $D3 = NV + NF + NIF + 3 * NP$

   and NV, NPV, NF, NIF, NP denote the total number of analysis variables, number of principal variables, number of factors to be computed, number of factors to be ignored, maximum number of points to be represented in the plots respectively.

26.9 Examples

Example 1. Factor analysis of correlations; analyses are based upon 20 variables and 7 factors are requested; number of factors to be rotated is defined according to the Kaiser criteria; statistics, correlation matrix and eigenvectors will be printed, followed by variable factors and standard plots; factors will not be kept in a file.

```
$RUN FACTOR
$FILES
PRINT  = FACT1.LST
DICTIN = A.DIC                    input Dictionary file
DATAIN = A.DAT                    input Data file
$SETUP
FACTOR ANALYSIS OF CORRELATIONS
ANAL=(NOCRSP,CORR)  ROTA=KAISER  NFACT=7  IDVAR=V1  PRINT=(STATS,MATRIX)  -
    PVARS=(V12-V16,V101-V115)
```

Example 2. Factor analysis of scalar products based upon 10 variables; 2 supplementary variables, V5 and V7, are to be represented on plots; plots are defined by user since only the 1st point of overlapping points is required; Kaiser's criteria are used to determine the number of factors; both variable and case factors will be written into files.

```
$RUN FACTOR
$FILES
DICTIN   = A.DIC                  input Dictionary file
DATAIN   = A.DAT                  input Data file
DICTOUT  = CASEF.DIC              Dictionary file for case factors
DATAOUT  = CASEF.DAT              Data file for case factors
DICTOUTV = VARF.DIC               Dictionary file for variable factors
DATAOUTV = VARF.DAT               Data file for variable factors
$SETUP
FACTOR ANALYSIS OF SCALAR PRODUCTS
ANAL=(NOCRSP,SSPR)  IDVAR=V1  WRITE=(OBSERV,VARS) PRINT=STATS  PLOT=USER  -
   PVARS=(V112-V116,V201-V205)  SVARS=(V5,V7)
X=1  Y=2  VARP=(PRINCIPAL,SUPPL)
X=1  Y=3  VARP=(PRINCIPAL,SUPPL)
X=2  Y=3  VARP=(PRINCIPAL,SUPPL)
```

Example 3. Correspondence analyses using a contingency table described by a dictionary and entered as a dataset in the Setup file to be executed; number of factors is defined by the Kaiser's criterion; matrix of relations will be printed, followed by variable and case factors, and by user defined plots of variables and cases.

```
$RUN FACTOR
$FILES
PRINT  = FACT3.LST
$SETUP
CORRESPONDENCE ANALYSIS ON CONTINGENCY TABLE
BADD=MD1  IDVAR=V8  PLOTS=USER  PRINT=(MATRIX,OFPRINC) PVARS=(V31-V33)
$DICT
$PRINT
   3   8  33   1   1
T   8 Scientific degree          1   20
C   8         81     Professor
C   8         82     Ass.Prof.
C   8         83     Doctor
C   8         84     M.Sc
C   8         85     Licence
C   8         86     Other
T  31 Head                       4   20
T  32 Scientifc                  7   20
T  33 Technician                10   20
$DATA
$PRINT
81  5  0  0
82  1  3  0
83  0 17 01
84  0 28 04
85  0  0 01
86  0  0 17
```

Chapter 27

Linear Regression (REGRESSN)

27.1 General Description

REGRESSN provides a general multiple regression capability designed for either standard or stepwise linear regression analysis. Several regression analyses, using different parameters and variables, may be performed in one execution.

Constant term. If the input is raw data, the user may request that the equations have no constant term (see the regression parameter CONSTANT=0). In such case, a matrix based on the cross-product matrix is analyzed instead of a correlation matrix. This changes the slope of the fitted line and can substantially affect the results. In stepwise regression, variables may enter the equation in a different order than they would if a constant term were estimated. If a correlation matrix is input, the regression equation always includes a constant term.

Use of categorical variables as independent variables. An option is available to create a set of dummy (dichotomous) variables from specified categorical variables (see the parameter CATE). These can be used as independent variables in the regression analysis.

F-ratio for a variable to enter in the equation. In a stepwise regression, variables are added in turn to the regression equation until the equation is satisfactory. At each step the variable with the highest partial correlation with the dependent variable is selected. A partial F-test value is then computed for the variable and this value is compared to a critical value supplied by the user. As soon as the partial F for the next to be entered variable becomes less than the critical value, the analysis is terminated.

F-ratio for a variable to be removed from the equation. A variable which may have been the best single variable to enter at an early stage of a stepwise regression may, at a later stage, not be the best because of the relationship between it and other variables now in the regression. To detect this, the partial F-value for each variable in the regression at each step of the calculation is computed and compared with a critical value supplied by the user. Any variable whose partial F-value falls below the critical value is removed from the model.

Stepwise regression. If stepwise regression is requested, the program determines which variables or which sets of dummy variables among the specified set of independent variables will actually be used for the regression, and in which order they will be introduced, beginning with the forced variables and continuing with the other variables and sets of dummy variables, one by one. After each step the algorithm selects from the remaining predictor variables the variable or set of dummy variables which yields the largest reduction in the residual (unexplained) variance of the dependent variable, unless its contribution to the total F-ratio for the regression remains below a specified threshold. Similarly, the algorithm evaluates after each step whether the contribution of any variable or set of dummy variables already included falls below a specified threshold, in which case it is dropped from the regression.

Descending stepwise regression. Like the stepwise regression, except that the algorithm starts with all the independent variables and then drops variables and sets of dummy variables in a stepwise manner. At each step the algorithm selects from the remaining included predictor variables the variable or set of dummy variables which yields the smallest reduction in the explained variance of the dependent variable, unless this exceeds a specified threshold. Similarly, the algorithm evaluates at each step whether the contribution of

any variable or set of dummy variables previously dropped from the regression has risen above a specified threshold, in which case it is added back into the regression.

Generating a residuals dataset. With raw data input, residuals may be computed and output as a data file described by an IDAMS dictionary. See the "Output Residuals Datasets" section for details on the content. Note that a separate residuals dataset is generated from each equation. Also, since REGRESSN has no facility to transfer specific variables of interest in a residuals analysis from the input raw data to the residuals dataset, it may be necessary to use the MERGE program to create the dataset containing all of the desired variables. A case ID variable from the input dataset is output to the residuals dataset to make matching possible.

Generating a correlation matrix. If raw data are input, the program computes correlation coefficients which may be output in the format of an IDAMS square matrix and used for further analysis. REGRESSN correlations include all variables across all regression equations and are based on cases which have valid data on all variables in the matrix. Thus, the correlations will usually differ from correlations obtained from the PEARSON program execution with the MDHANDLING=PAIR option. When missing data elimination in REGRESSN leaves the sample size acceptably large, REGRESSN is an alternative to PEARSON for generating a correlation matrix (see the paragraph "Treatment of missing data").

27.2 Standard IDAMS Features

Case and variable selection. If raw data are input, the standard filter is available to select a subset of cases from the input data. If a matrix of correlations is used as input to the program, case selection is not applicable. The variables for the regression equation are specified in the regression parameters DEPVAR and VARS.

Transforming data. If raw data are input, Recode statements may be used.

Weighting data. If raw data are input, a variable can be used to weight the input data; this weight variable may have integer or decimal values. The program will force the sum of the weights to equal the number of input cases. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data.

1. **Input.** If raw data are input, the MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases in which missing data occur in any regression variable in any analysis are deleted ("case-wise" missing data deletion). An option (see the parameter MDHANDLING) allows the user to specify the maximum number of missing data cases which can be tolerated before the execution is terminated. Warning: If multiple analyses are performed in one REGRESSN execution, a single correlation matrix is computed for all variables used in the different analyses. Because of the "case-wise" method of deleting cases with missing data, the number of cases used and thus the regression statistics produced may be different if the analyses are then performed separately.

   If a matrix is input, cases with missing data should have been accommodated when the matrix was created. If a cell of the input matrix has a missing data code (i.e. 99.999) any analysis involving that cell will be skipped.

2. **Output residuals.** If residuals are requested, predicted values and residuals are computed for all cases which pass the (optional) filter. If a case has missing data on any of the variables required for these computations, output missing data codes are generated.

3. **Output correlation matrix.** The REGRESSN algorithm for handling missing data on raw data input cannot result in missing data entries in the correlation matrix.

27.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Univariate statistics. (Raw data input only). The sum, mean, standard deviation, coefficient of variation, maximum, and minimum are printed for all dependent and independent variables used.

Matrix of total sums of squares and cross-products. (Raw data input only. Optional: see the parameter PRINT).

Matrix of residual sums of squares and cross-products. (Raw data input only. Optional: see the parameter PRINT).

Total correlation matrix. (Optional: see the parameter PRINT).

Partial correlation matrix. (Optional for each regression: see the regression parameter PARTIALS). The ij-th element is the partial correlation between variable i and variable j, holding constant the variables specified in the PARTIALS variable list.

Inverse matrix. (Optional for each regression: see the regression parameter PRINT).

Analysis summary statistics. The following statistics are printed for each regression or for each step of a stepwise regression:

standard error of estimate,
F-ratio,
multiple correlation coefficient (adjusted and unadjusted),
fraction of explained variance (adjusted and unadjusted),
determinant of the correlation matrix,
residual degrees of freedom,
constant term.

Analysis statistics for predictors. The following statistics are printed for each regression or for each step of a stepwise regression:

coefficient B (unstandardized partial regression coefficient),
standard error (sigma) of B,
coefficient beta (standardized partial regression coefficient),
standard error (sigma) of beta,
partial and marginal R squared,
t-ratio,
covariance ratio,
marginal R squared values for all predictors and t-ratios for all sets of dummy variables (for stepwise regression).

Residual output dictionary. (For raw data input only. Optional: see the regression parameter WRITE).

Residual output data. (For raw data input only. Optional: see the regression parameter PRINT). If there are less than 1000 cases, calculated values, observed values and residuals (differences) may be listed in ascending order of residual value. Any number of cases may be listed in input case sequence order. The Durbin-Watson statistic for association of residuals will be printed for residuals listed in case sequence order.

27.4 Output Correlation Matrix

The computed correlation matrix may be output (see the parameter WRITE). It is written in the form of an IDAMS square matrix (see "Data in IDAMS" chapter). The format is 6F11.7 for the correlations and 4E15.7 for the means and standard deviations. In addition, labeling information is written in columns 73-80 of the records as follows:

matrix-descriptor record	N=nnnnn
correlation records	REG xxx
means records	MEAN xxx
standard deviation records	SDEV xxx

(nnnnn is the REGRESSN sample size. The xxx is a sequence number beginning with 1 for the first correlation record and incremented by one for each successive record through the last standard deviation record).

The elements of the matrix are Pearson r's. They, as well as the means and standard deviations, are based on the cases that have valid data on all the variables specified in any of the regression variable lists. The correlations are for all pairs of variables from all the analysis variable lists taken together.

27.5 Output Residuals Dataset(s)

For each analysis, a residuals dataset can be requested (see the regression parameter WRITE). This is output in the form of a Data file described by an IDAMS dictionary. It contains either four or five variables per case, depending on whether or not the data were weighted: an ID variable, a dependent variable, a predicted (calculated) dependent variable, a residual, and a weight, if any. Cases are output in the order of the input cases. The characteristics of the dataset are as follows:

	Variable No.	Name	Field Width	No. of Decimals	MD1 Code
(ID variable)	1	same as input	*	0	same as input
(dependent variable)	2	same as input	*	**	same as input
(predicted variable)	3	Predicted value	7	***	9999999
(residual)	4	Residual	7	***	9999999
(weight-if weighted)	5	same as input	*	**	same as input

*	transferred from input dictionary for V variables or 7 for R variables
**	transferred from input dictionary for V variables or 2 for R variables
***	6 plus no. of decimals for dependent variable minus width of dependent variable; if this is negative, then 0.

If the calculated value or residual exceeds the allocated field width, it is replaced by MD1 code.

27.6 Input Dataset

The input raw dataset is a Data file described by an IDAMS dictionary. All variables used for analysis must be numeric; they may be integer or decimal valued. The case ID variable can be alphabetic.

27.7 Input Correlation Matrix

This is an IDAMS square matrix. A correlation matrix generated by PEARSON or by a previous REGRESSN is an appropriate input matrix for REGRESSN.

The input matrix dictionary must contain variable numbers and names. The matrix must contain correlations, means and standard deviations. Both the means and standard deviations are used.

27.8 Setup Structure

```
$RUN REGRESSN

$FILES
     File specifications

$RECODE (optional with raw data input; unavailable with matrix input)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Definition of dummy variables (conditional)
     5. Regression specifications (repeated as required)

$DICT (conditional)
     Dictionary for raw data input

$DATA (conditional)
     Data for raw data input

$MATRIX (conditional)
     Matrix for correlation matrix input


Files:
FT02       output correlation matrix
FT09       input correlation matrix
           (if $MATRIX not used and INPUT=MATRIX)
DICTxxxx   input dictionary (if $DICT not used and INPUT=RAWDATA)
DATAxxxx   input data (if $DATA not used and INPUT=RAWDATA)
DICTyyyy   output residuals distionary )  one set for each
DATAyyyy   output residuals data       )  residuals file requested
PRINT      results (default  IDAMS.LST)
```

27.9 Program Control Statements

Refer to "The IDAMS setup file" chapter for further descriptions of the program control statements, items 1-3 and 5 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution. Available only with raw data input.

   ```
   Example:  INCLUDE V3=5
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  REGRESSION ANALYSIS
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  IDVAR=V1  MDHANDLING=100
   ```

INPUT=**RAWDATA**/MATRIX

RAWD The input data are in the form of a Data file described by an IDAMS dictionary.
MATR The input data are correlation coefficients in the form of an IDAMS square matrix.

Parameters only for raw data input

INFILE=**IN**/xxxx

A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2

Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n

The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE

Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=**0**/n

The number of missing data cases to be allowed before termination. A case is counted missing if it has missing data in any of the variables in the regression equations.

WEIGHT=variable number

The weight variable number if the data are to be weighted.

CATE

Specify CATE if a definition of dummy variables is provided.

IDVAR=variable number

Variable to be output or printed as case ID if residuals dataset is requested. The ID variable should not be included in any variable list.

WRITE=MATRIX

Write the correlation matrix computed from the raw data input to an output file.

PRINT=(CDICT/DICT, XMOM, XPRODUCTS, MATRIX)

CDIC Print the input dictionary for the variables accessed with C-records if any.
DICT Print the input dictionary without C-records.
XMOM Print the matrix of residual sums of squares and cross-products.
XPRO Print the matrix of total sums of squares and cross-products.
MATR Print the correlation matrix.

Parameters for correlation matrix input

CASES=n

Set CASES equal to the number of cases used to create the input matrix. This number is used in calculating the F-level.
No default; must be supplied when correlation matrix input.

PRINT=MATRIX

Print the correlation matrix.

4. **Definition of dummy variables** (conditional: if CATE was specified as a parameter). The REGRESSN program can transform a categorical variable to a set of dummy variables. To have a variable

treated as categorical, the user must a) include the CATE parameter in the parameter list and b) specify the variables to be considered categorical and the codes to be used. Each categorical variable to be transformed is followed by the codes to be used enclosed in brackets. For each variable, any codes not listed will be excluded from the construction. Note: The list of codes should not be exhaustive, i.e. all existing codes should not be listed or else a singular matrix will result.

```
Example:  V100(5,6,1), V101 (1-6)
```

Codes 5, 6 and 1 of variable 100 will be represented in the regression as dummy variables, along with codes 1 through 6 of variable 101.

A variable specified in the definition of dummy variables, when used in predictor (VARS), partials (PARTIALS) or forced (FORCE) variables lists for stepwise regression, will refer to the set of dummy variables created from that variable. In stepwise regressions, the codes of such a variable will be entered or excluded together, and marginal R-squares and F-ratios will be calculated for all codes of the variable together as well as for codes individually. A variable used in a definition of dummy variables may not be used as a dependent variable.

5. **Regression specifications.** The coding rules are the same as for parameters. Each set of regression parameters must begin on a new line.

```
Example:  DEPV=V5  METH=STEP  FORCE=(V7) VARS=(V7,V16,V22,V37-V47,R14)
```

METHOD=**STANDARD**/STEPWISE/DESCENDING

STAN	A standard regression will be done.
STEP	A stepwise regression will be done.
DESC	A descending stepwise regression will be done.

DEPVAR=variable number
Variable number of dependent variable.
No default.

VARS=(variable list)
The independent variables to be used in this analysis.
No default.

PARTIALS=(variable list)
Compute and print a partial correlation matrix with the specified variables removed from the independent variable list.
Default: No partials.

FORCE=(variable list)
Force the variables listed to enter into the stepwise regression (METH=STEP) or to remain in the descending stepwise regression (METH=DESC).
Default: No forcing.

FINRATIO=**.001**/n
The F-ratio value below which a variable will not be entered in a stepwise procedure; this is the F-ratio to enter. The decimal point must be entered.

FOUTRATIO=**0.0**/n
The F-ratio value above which a variable must remain in order to continue in a stepwise procedure; this is the F-ratio to remove. The decimal point must be entered.

CONSTANT=0
For raw data input only.
The constant term is required to equal zero and no constant term will be estimated.
Default: A constant term will be estimated.

WRITE=RESIDUALS
Residuals are to be written out as an IDAMS dataset.

OUTFILE=**OUT**/yyyy
Applicable only if WRITE=RESI specified.
A 1-4 character ddname suffix for the residuals output Dictionary and Data files. If outputting residuals from more than 1 analysis, the default ddname, OUT, may be used only once.

PRINT=(STEP, RESIDUALS, ERESIDUALS, INVERSE)

STEP	Applies to the stepwise regression only: print marginal R-squares for all predictors in each step.
RESI	Print residuals in input case sequence order and Durbin-Watson statistic.
ERES	Print residuals, except for missing data, in error magnitude order, provided there are fewer than 1000 cases.
INVE	Print the inverse correlation matrix.

27.10 Restrictions

1. With raw data input, there may be as many as 99 or 100 (depending on whether a weight variable is used) distinct variables used in any single regression equation; the total number of variables across all analysis, including Recode variables, weight variable and ID variable, can be no more than 200.
2. With matrix input, the matrix can be 200 x 200, and up to 100 variables may be used in any single regression equation.
3. FINRATIO must be greater than or equal to FOUTRATIO.
4. Residuals may be listed in ascending order of residual value only if there are fewer than 1000 cases.
5. A variable specified in a definition of dummy variables may not be used as a dependent variable.
6. Maximum 12 dummy variables can be defined from one categorical variable.
7. If the ID variable is alphabetic with width > 4, only the first four characters are used.

27.11 Examples

Example 1. Standard regression with five independent variables using an IDAMS correlation matrix as input.

```
$RUN REGRESSN
$FILES
FT09 = A.MAT                          input Matrix file
SETUP
STANDARD REGRESSION  -  USING MATRIX AS INPUT
INPUT=MATR  CASES=1460
DEPV=V116  VARS=(V18,V36,V55-V57)
```

Example 2. Standard regression with six independent variables and with two variables each with 3 categories transformed to 6 dummy variables; raw data are used as input; residuals are to be computed and written into a dataset (cases are identified by variable V2).

```
$RUN REGRESSN
$FILES
PRINT   = REGR2.LST
DICTIN  = STUDY.DIC                   input Dictionary file
DATAIN  = STUDY.DAT                   input Data file
```

```
DICTOUT = RESID.DIC                    Dictionary file for residuals
DATAOUT = RESID.DAT                    Data file for residuals
$SETUP
STANDARD REGRESSION  -  USING RAW DATA AS INPUT AND WRITING RESIDUALS
MDHANDLING=50  IDVAR=V2  CATE
V5(1,5,6),V6(1-3)
DEPV=V116  WRITE=RESI  VARS=(V5,V6,V8,V13,V75-V78)
```

Example 3. Two regressions: one standard and one stepwise using raw data as input.

```
$RUN REGRESSN
$FILES
DICTIN = STUDY.DIC                     input Dictionary file
DATAIN = STUDY.DAT                     input Data file
$SETUP
TWO REGRESSIONS
PRINT=(XMOM,XPROD)
DEPV=V10  VARS=(V101-V104,V35)  PRINT=INVERSE
DEPV=V11  METHOD=STEP  PRINT=STEP VARS=(V1,V3,V15-V18,V23-V29)
```

Example 4. Two-stage regression; the first stage uses variables V2-V6 to estimate values of the dependent variable V122; in the 2nd stage, two additional variables V12, V23 are used to estimate the predicted values of V122, i.e. V122 with the effects of V2-V6 removed.

In the first regression, predicted values for the dependent variable (V122) are computed and written to the residuals file (OUTB) as variable V3. MERGE is then used to merge this variable with the variables from the original file that are required in the second stage. The output dataset from MERGE (a temporary file so it need not be defined) will contain the 5 variables from the build list, numbered V1 to V5 where A12 and A23 (to be used as predictors in the second stage) become V2 and V3, A122, the original dependent variable, becomes V4, and B3, the variable giving predicted values of V122 becomes V5. This output file is then used as input to the second stage regression.

```
$RUN REGRESSN
$FILES
PRINT    = REGR4.LST
DICTIN   = STUDY.DIC                   input Dictionary file
DATAIN   = STUDY.DAT                   input Data file
DICTOUTB = RESID.DIC                   Dictionary file for residuals
DATAOUTB = RESID.DAT                   Data file for residuals
$SETUP
TWO STAGE REGRESSION  -  FIRST STAGE
MDHANDLING=100  IDVAR=V1
DEPV=V122  WRITE=RESI  OUTF=OUTB  VARS=(V2-V6)
$RUN MERGE
$SETUP
MERGING PREDICTED VALUE (V3 IN RES FILE) INTO DATA FILE
MATCH=INTE  INAF=IN  INBF=OUTB
A1=B1
A1,A12,A23,A122,B3
$RUN REGRESSN
$SETUP
TWO STAGE REGRESSION  -  SECOND STAGE
MDHANDLING=100  INFI=OUT
DEPV=V5  VARS=(V2,V3)
```

Chapter 28

Multidimensional Scaling (MDSCAL)

28.1 General Description

MDSCAL is a non-metric multidimensional scaling program for the analysis of similarities. The program, which operates on a matrix of similarity or dissimilarity measures, is designed to find, for each dimensionality specified, the best geometric representation of the data in the space.

The uses of non-metric multidimensional scaling are similar to those of factor analysis, e.g. clusters of variables can be spotted, the dimensionality of the data can be discovered, and dimensions can sometimes be interpreted. The CONFIG program can be used to perform analysis on an MDSCAL output configuration.

Input configuration. Normally an internally created arbitrary starting configuration is used to begin the computation. The user may, however, supply an initial configuration. There are several possible reasons for providing a starting configuration. The user may have theoretical reasons for beginning with a certain configuration; one may wish to perform further iteration on a configuration which is not yet close enough to the best configuration; or, to save computing time, one may wish to provide a higher dimensional configuration as a starting point for a lower dimensional configuration.

Scaling algorithm. The program starts with an initial configuration, either generated arbitrarily or supplied by the user, and iterates (using a procedure of the "steepest descent" type) over successive trial configurations, each time comparing the rank order of inter-point differences in the trial configuration with the rank order of the corresponding measure in the data. A "badness of fit" measure (stress coefficient) is computed after each iteration and the configuration is rearranged accordingly to improve the fit to the data, until, ideally, the rank order of distances in the configuration is perfectly monotonic with the rank order of dissimilarities given by the data; in that case, the "stress" will be zero. In practice, the scaling computation stops, in any given number of dimensions, because the stress reaches a sufficiently small value (STRMIN), the scale factor (magnitude) of the gradient reaches a sufficiently small value (SRGFMN), the stress has been improving too slowly (SRATIO), or the preset maximum number of iterations is reached (ITERATIONS). The program stops on whichever condition comes first. The same procedure is repeated for the next lower dimensionality using the previous results as the initial configuration, until a specified minimum number of dimensions is reached. During computation, the cosine of the angle between successive gradients plays an important role in several ways; optionally, two internal weighting parameters may be specified (see parameters COSAVW and ACSAVW).

Dimensionality and metric. Solutions may be obtained in 2 to 10 dimensions. The user controls the dimensionality of the configurations obtained by specifying the maximum and minimum number of dimensions desired, and the difference between the dimensionality of the successive solutions produced (see parameters DMAX, DMIN, and DDIF). The user also specifies, using parameter R, whether the distance metric should be Euclidean (R=2), the usual case, or some other Minkowski r-metric.

Stress. Stress is a measure of how well the configuration matches the data. The user may choose between two alternate formulas for computing the stress coefficient: either the stress is standardized by the sum of the squared distances from the mean (SQDIST) or the stress is standardized by the sum of the squared deviations from the mean (SQDEV). In many situations, the configurations reached by the two formulas will not be substantially different. Larger values of stress result from formula 2 for the same degree of fit.

Ties in input coefficients. There are two alternative methods for handling ties among the input data values; the corresponding distances can be required to be equal (TIES=EQUAL) or they can be allowed to differ (TIES=DIFFER). When there are few ties, it makes little difference which approach is used. When there are a great many ties it does make a difference, and the context must be considered in making the choice.

28.2 Standard IDAMS Features

Case and variable selection. Filtering of cases must be performed at the time the matrix is created, not in MDSCAL. The parameter VARS allows the computation to be performed on subsets of the matrix rather than on the entire matrix.

Transforming data. Use of Recode statements is not applicable in MDSCAL. Data transformations must be performed at the time the input matrix is created.

Weighting data. Weighting in the usual sense (weighting cases to correct for different sampling rates or different levels of aggregation) must be accomplished before using MDSCAL; such weighting must be incorporated in the input data matrix. There is a weight option of a quite different sort available in MDSCAL (see parameter INPUT=WEIGHTS). It may be used to assign weights to cells of the input matrix; the user supplies a matrix of values which are to be used as weights for the corresponding elements in the input matrix.

Treatment of missing data. Missing data for individual cases must be accounted for at the time the input data matrix is created, not in MDSCAL. If, after the matrix has been created, an entry in the matrix is missing, i.e. contains a missing data code, there is a possibility of processing it in MDSCAL: the MDSCAL cutoff option (see parameter CUTOFF) can be used to exclude from analysis missing data values if these are less than valid data values. MDSCAL has no option for recognizing missing data values that are large numbers (such as 99.99901, the missing data code output by PEARSON). If large missing data values do exist, these should be edited to small numbers. If one particular variable has many missing entries, possibly it should be dropped from the analysis.

28.3 Results

Input matrix. (Optional: see the parameter PRINT).

Input weights. (Optional: see the parameter PRINT).

Input configuration. If a starting configuration is supplied, it is always printed.

History of the computation. For each solution, the program prints a complete history of computations, reporting the stress value and its ancillary parameters for each iteration:

Iteration	the iteration number
Stress	the current value of the stress
SRAT	the current value of the stress ratio
SRATAV	the current stress ratio average (it is an exponentially weighted average)
CAGRGL	the cosine of the angle between the current gradient and the previous gradient
COSAV	the current value of the average cosine of the angle between successive gradients (a weighted average)
ACSAV	the current value of the average absolute value of the cosine of the angle between successive gradients (a weighted average)
SFGR	the length (more properly, the scale factor) of the gradient
STEP	the step size.

Reason for termination. When computation is terminated, the reason is indicated by one of the remarks: "Minimum was achieved", "Maximum number of iterations were used", "Satisfactory stress was reached", or "Zero stress was reached".

Final configuration. For each solution, the Cartesian coordinates of the final configuration are printed.

Sorted configuration. (Optional: see the parameter PRINT). For each solution, the projections of points of the final configuration are sorted separately on each dimension into ascending order and printed.

Summary. For each solution, the original data values are sorted and printed together with their corresponding final distances (DIST) and the hypothetical distances required for a perfect monotonic fit (DHAT).

28.4 Output Configuration Matrix

As the final configuration for each dimensionality is calculated, it may be output as an IDAMS rectangular matrix. The configuration is centered and normalized. The rows represent variables and the columns represent dimensions. The matrix elements are written in 10F7.3 format. Dictionary records are generated. This matrix may be submitted as a configuration input for another execution of MDSCAL or it may be input to another program such as CONFIG for additional analysis.

28.5 Input Data Matrix

The usual input to MDSCAL is an IDAMS square matrix (see "Data in IDAMS" chapter). This matrix is the upper-right-half matrix with no diagonal and it is defined by the parameter INPUT=STANDARD. TABLES and PEARSON generate matrices suitable for input to MDSCAL. Means and standard deviations are not used but appropriate (dummy) records must be supplied. MDSCAL will accept matrices in other formats than the upper-right triangle with no diagonal. However, such matrices must contain the dictionary portion of an IDAMS square matrix and must have records containing pseudo means and standard deviations at the end.

The following INPUT parameters indicate the exact format of matrix being input:

STAN	upper-right triangle, no diagonal
STAN, DIAG	upper-right triangle, with diagonal
LOWER, DIAG	lower-left triangle, with diagonal
LOWER	lower-left triangle, no diagonal
SQUARE	full square matrix with diagonal.

The measures contained in the data matrix may either be measures of similarity (such as correlations) or dissimilarities. Although the input to MDSCAL is usually a matrix of correlation coefficients (e.g. a matrix of gammas or a matrix of Pearson r's), the input matrix may contain any measure that makes sense as a measure of proximity. Because non-metric scaling uses only ordinal properties of the data, nothing need be assumed about the quantitative or numerical properties of the data. There should be, at the very least, twice as many variables as dimensions.

28.6 Input Weight Matrix

If a weight matrix is supplied, it must be in exactly the same format as the input data matrix. The parameter INPUT=(STAN/LOWE/SQUA, DIAG) applies to the weight matrix as well as to the data matrix. The dictionary for the weight matrix should be the same as for the input data matrix. Means and standard deviations are not used, but corresponding "dummy" lines should be supplied.

This matrix contains values, in one-to-one correspondence with elements of the data matrix, which are to be used as weights for the data. These values are used in conjunction with the value for the parameter CUTOFF when applied to the data. If a data value is greater than the cutoff value, but the corresponding weight value is less than or equal to zero, an error condition is signaled. Likewise, if the data value is less than or equal to the cutoff value, and the corresponding weight value is greater than zero, an error condition is set. If either of these inconsistencies occurs, the execution terminates.

28.7 Input Configuration Matrix

The input configuration must be in the format of an IDAMS rectangular matrix. See "Data in IDAMS" chapter.

It provides a starting configuration to be used in the computations. The rows should represent variables and the columns dimensions. It is usually produced by a previous execution of MDSCAL and is submitted in order that a previous execution may start where it left off.

The matrix must contain at least as many dimensions as the value given for the parameter DMAX.

Note: If a variable list (VARS) is specified, MDSCAL uses the **first** n rows of the input configuration where n is the number of variables in the list, without checking the variable numbers.

28.8 Setup Structure

```
$RUN MDSCAL

$FILES
     File specifications

$SETUP
     1. Label
     2. Parameters

$MATRIX (conditional)
     Data matrix
     Weight matrix
     Starting configuration matrix
(Note:  Not all of the matrices need be included here; however, if
 more than one matrix is included, they must be in the above order).

Files:
FT02    output configuration matrix
FT03    input weight matrix if INPUT=WEIGHTS specified (omit if $MATRIX used)
FT05    input starting configuration if INPUT=CONFIG specified
        (omit if $MATRIX used)
FT08    input data matrix (omit if $MATRIX used)
PRINT   results (default  IDAMS.LST)
```

28.9 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-2 below.

1. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  MDSCAL EXECUTION ON DATASET X4952
   ```

2. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  DMAX=5  ITER=75  WRITE=CONFIG
   ```

INPUT=(**STANDARD**/LOWER/SQUARE, DIAGONAL, WEIGHTS, CONFIG)

STAN	The input is an IDAMS square matrix, i.e. off-diagonal, upper-right-half matrix.
LOWE	The input matrix is a lower-left-half matrix.
SQUA	The input matrix is a full square matrix.
DIAG	The input matrix has the diagonal elements.
WEIG	A matrix of weight values is being supplied.
CONF	The starting configuration matrix is being supplied.

VARS=(variable list)

List of variables in the matrix on which analysis is to be performed.
Default: The entire input matrix is used.

FILE=(DATA, WEIGHTS, CONFIG)

DATA	The input data matrix is in a file.
WEIG	The weight matrix is in a file.
CONF	The input configuration matrix is in a file.

Default: All matrices are assumed to follow a $MATRIX command in the order data, weight, configuration.

COEFF=**SIMILARITIES**/DISSIMILARITIES

SIMI	Large coefficients in the data matrix indicate that points are similar or close.
DISS	Large coefficients indicate that points are dissimilar or far.

DMAX=**2**/n

The dimension maximum: scaling starts with the space of maximum dimension.

DMIN=**2**/n

The dimension minimum: scaling proceeds until it reaches or would pass the minimum dimension.

DDIF=**1**/n

The dimension difference: scaling proceeds from maximum dimension to minimum dimension by steps of the dimension difference.

R=**2.0**/n

Indicate which Minkowski r-metric is to be used. Any value >= 1.0 can be used.

R=1.0	City block metric.
R=2.0	Ordinary Euclidean distance.

CUTOFF=**0.0**/n

Data values less than or equal to n are discarded. If the legitimate values of the input coefficients range from -1.0 to 1.0, CUTOFF=-1.01 should be used.

TIES=**DIFFER**/EQUAL

DIFF	Unequal distances corresponding to equal data values do not contribute to the stress coefficient and no attempt is made to equalize these distances.
EQUA	Unequal distances corresponding to equal data values do contribute to the stress and there is an attempt to equalize these distances.

ITERATIONS=**50**/n

The maximum number of iterations to be performed in any given number of dimensions. This maximum is a safety precaution to control execution time.

STRMIN=**.01**/n

Stress minimum. The scaling procedure will stop if the stress reaches the minimum value.

SFGRMN=**0.0**/n
Minimum value of the scale factor of the gradient. The scaling procedure will stop if the magnitude of the gradient reaches the minimum value.

SRATIO=**.999**/n
The stress ratio. Scaling procedure stops if the stress ratio between successive steps reaches n.

ACSAVW=**.66**/n
The weighting factor for the average absolute value of the cosine of the angle between successive gradients.

COSAVW=**.66**/n
The weighting factor for the average cosine of the angle between successive gradients.

STRESS=**SQDIST**/SQDEV
SQDI Compute the stress using the standardization by the sum of the squared distances.
SQDE Compute the stress using the standardization by the sum of the squared deviations from the mean.

WRITE=CONFIG
Output the final configuration of each solution into a file.

PRINT=(MATRIX, SORTCONF, **LONG**/SHORT)
MATR Print the input data matrix and the weight matrix if one is supplied.
SORT Sort each dimension of the final configuration and print it.
LONG Print matrices on long lines.
SHOR Print matrices on short lines.

28.10 Restrictions

1. The capacity of the program is 1800 data points (e.g. 1800 elements of the similarity or dissimilarity matrix). This is equivalent to a triangle of a 60 x 60 matrix or to a 42 x 42 square matrix.
2. Variables may be scaled in up to 10 dimensions.
3. The starting configuration matrix may have a maximum of 60 rows and 10 columns.

28.11 Example

Generation of an output configuration matrix; the input data matrix is in standard IDAMS form and in a file; there is neither input weight matrix nor input configuration matrix; 20 iterations are requested; analysis is to be performed on a subset of variables.

```
$RUN MDSCAL
$FILES
FT02 = MDS.MAT                  output configuration Matrix file
FT08 = ABC.COR                  input data Matrix file
$SETUP
MULTIDIMENSIONAL SCALING
ITER=20  WRITE=CONFIG  FILE=DATA  VARS=(V18-V36)
```

Chapter 29

Multiple Classification Analysis (MCA)

29.1 General Description

MCA examines the relationships between several predictor variables and a single dependent variable and determines the effects of each predictor before and after adjustment for its inter-correlations with other predictors in the analysis. It also provides information about the bivariate and multivariate relationships between the predictors and the dependent variable. The MCA technique can be considered the equivalent of a multiple regression analysis using dummy variables. MCA, however, is often more convenient to use and interpret. MCA also has an option for one-way analysis of variance.

MCA assumes that the effects of the predictors are additive i.e. that there are no interactions between predictors. It is designed for use with predictor variables measured on nominal, ordinal, and interval scales. It accepts an unequal number of cases in the cells formed by cross-classification of the predictors.

Alternatives to MCA are REGRESSN and ONEWAY. REGRESSN provides a general multiple regression capability. ONEWAY performs a one-way analysis of variance. The advantage of MCA over REGRESSN is that it accepts predictor variables in as weak a form as nominal scales, and it does not assume linearity of the regression. The advantages over ONEWAY are that in MCA the maximum code for a control variable in a one-way analysis is 2999 (instead of 99 in ONEWAY).

Generating a residuals dataset. Residuals may be computed and output as a Data file described by an IDAMS dictionary. See the "Output Residuals Dataset(s)" section for details on the content. The option is not available if only one predictor is specified.

Iterative procedures. MCA uses an iteration algorithm for approximating the coefficients constituting the solutions to the set of normal equations. The iteration algorithm stops when the coefficients being generated are sufficiently accurate. This involves setting a tolerance and specifying a test for determining when that tolerance has been met (see analysis parameters CRITERION and TEST). Four convergence tests are available. If the coefficients do not converge within the limits set by the user, the program prints out its results on the basis of the last iteration. The number of useful iterations depends somewhat on the number of predictors used in the analysis and on the fraction specified for tolerance. If there are fewer than 10 predictors, it has usually been found satisfactory to specify 10 as the maximum number of iterations.

Detection and treatment of interactions. The program assumes that the phenomena being examined can be understood in terms of an additive model.

If, on a priori grounds, particular variables are suspected to be interacting, MCA itself can be used to determine the extent of the interaction as follows. If one predictor is specified, MCA performs a one-way analysis of variance. Such an analysis can assist in detecting and eliminating predictor interactions. The complete procedure is as follows (see also Example 3):

1. Determine a set of suspected interacting predictors.
2. Form a single "combination variable" using these predictors and the Recode statement COMBINE.

3. Perform one MCA analysis using the suspect predictors to get adjusted R squared.
4. Perform one MCA analysis with the "combination variable" as the control in a one-way analysis of variance to get adjusted eta squared, which will be greater than or equal to adjusted R squared.
5. Use the difference, adjusted eta squared-adjusted R squared (the fraction of variance explained which is lost due to the additivity assumption), as a guide to determine whether the use of a combination variable in place of the original predictors is justified.

The test for interaction must be based on the same sample as the normal MCA execution. If interactions are detected, then the combination variable should be used as predictor variable in place of the individual interacting variables.

29.2 Standard IDAMS Features

Case and variable selection. Cases may be excluded from all analyses in the MCA execution by use of a standard filter statement. In multiple classification analysis, cases may be excluded also by exceeding the predictor maximum code. (Note: If a predictor variable from any analysis has a code outside the range 0-31, the case containing the value is eliminated from all analyses). For any particular analysis, additional cases may be excluded due to the following conditions:

- A case (referred to as an outlier) has a dependent variable value that is more than a specified number of standard deviations from the mean of the dependent variable. See analysis parameters OUTDISTANCE and OUTLIERS.
- A case has a dependent variable value that is greater than a specified maximum. See analysis parameter DEPVAR.
- A case has missing data for the dependent or weight variable. See the "Treatment of missing data" and "Weighting data" paragraphs below.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed. When weighted data are used, tests of statistical significance must be interpreted with caution.

Treatment of missing data. The MDVALUES analysis parameter is available to indicate which missing data values, if any, are to be used to check for missing data in the dependent variable. Cases with missing data in the dependent variable are always excluded. Cases with missing data in predictor variables may be excluded from all analyses using the filter. (Using the filter to exclude cases with missing data on predictor variables in multiple classification is only needed if the missing data codes are in the range 0-31; if the value for any predictor is outside this range, a case is automatically excluded from all analyses requested in the execution).

29.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Weighted frequency table. (Optional: see the analysis parameter PRINT). An N x M matrix is printed for each pair of predictors where N=maximum code of row predictor and M=maximum code of column predictor. The total number of tables is P(P-1)/2 where P is the number of predictors.

Coefficients for each iteration. (Optional: see the analysis parameter PRINT). The coefficients for each class for each predictor.

Dependent variable statistics. For the dependent variable (Y):
grand mean, standard deviation and coefficient of variation,
sum of Y and sum of Y-squared,
total, explained and residual sums of squares,
number of cases used in the analysis and sum of weights.

Predictor statistics for multiple classification analysis.

For each category of each predictor:
the category (class) code, and label if it exists in the dictionary,
the number of cases with valid data (in raw, weighted and per cent form),
mean (unadjusted and adjusted), standard deviation and coefficient of variation of the dependent variable,
unadjusted deviation of the category mean from the grand mean and, coefficient of adjustment.

For each predictor variable:
eta and eta squared (unadjusted and adjusted),
beta and beta squared,
unadjusted and adjusted sums of squares.

Analysis statistics for multiple classification analysis. For all predictors combined:
multiple R-squared (unadjusted and adjusted),
coefficient of adjustment for degrees of freedom,
multiple R (adjusted),
listing of betas in descending order of their values.

One-way analysis of variance statistics.

For each category of the predictor:
the category (class) code, and label if it exists in the dictionary,
the number of cases with valid data (in raw, weighted and per cent form),
mean, standard deviation and coefficient of variation of the dependent variable,
sum and percentage of dependent variable values,
sum of dependent variable values squared.

For the predictor variable:
eta and eta squared (unadjusted and adjusted),
coefficient of adjustment for degrees of freedom,
total, between means and within groups sums of squares,
F value (degrees of freedom are printed).

Residuals. (Optional: see the analysis parameter PRINT). The identifying variable, observed value, predicted value, residual and weight variable, if any, are printed for cases in the order of the input file.

Summary statistics of residuals. If residuals are requested, the program prints the number of cases, sum of weights, and mean, variance, skewness, and kurtosis of the residual variable.

29.4 Output Residuals Dataset(s)

For each analysis, residuals can optionally be output in a Data file described by an IDAMS dictionary. (See analysis parameter WRITE=RESIDUALS). A record is output for each case passing the filter containing an ID variable, an observed value, a calculated value, a residual value for the dependent variable and a weight variable value, if any. The characteristics of the dataset are as follows:

	Variable No.	Name	Field Width	No. of Decimals	MD Codes
(ID variable)	1	same as input	*	0	same as input
(dependent variable)	2	same as input	*	**	same as input
(predicted variable)	3	Predicted value	7	***	9999999
(residual)	4	Residual	7	***	9999999
(weight-if weighted)	5	same as input	*	**	same as input

*	transferred from input dictionary for V variables or 7 for R variables
**	transferred from input dictionary for V variables or 2 for R variables
***	6 plus no. of decimals for dependent variable minus width of dependent variable; if this is negative, then 0.

If the observed value or weight variable value is missing or the case was excluded by maximum code checking or by the outlier criteria, a residual record is output with all variables (except the identifying variable) set to MD1.

29.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. All variables used for analysis must be numeric; they may be integer or decimal valued, except for predictors which must have integer values, between 0 and 31 for multiple classification and up to 2999 for one-way analysis of variance. The case ID variable can be alphabetic.

A large number of cases is necessary for an MCA analysis; a good rule of thumb is that the total number of categories (i.e. the sum of categories over all predictors) should not exceed 10% of the sample size.

The dependent variable must be measured on an interval scale or be a dichotomy, and it should not be badly skewed. Predictor variables for MCA must be categorized, preferably with not more than 6 categories. Although MCA is designed to handle correlated predictors, no two predictors should be so strongly correlated that there is perfect overlap between any of their categories. (If there is perfect overlap, recoding to combine categories or filtering to remove offending cases is necessary).

29.6 Setup Structure

```
$RUN MCA

$FILES
     File specificaitions

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Analysis specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output residuals distionary )  one set for each
DATAyyyy   output residuals data       )  residuals file requested
PRINT      results (default  IDAMS.LST)
```

29.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V6=2-6
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  TEST RUN FOR MCA
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example:  *
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   BADDATA=**STOP**/SKIP/MD1/MD2
   Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

   MAXCASES=n
   The maximum number of cases (after filtering) to be used from the input file.
   Default: All cases will be used.

   PRINT=CDICT/DICT
   CDIC Print the input dictionary for the variables accessed with C-records if any.
   DICT Print the input dictionary without C-records.

4. **Analysis specifications.** The coding rules are the same as for parameters. Each analysis specification must begin on a new line.

   ```
   Example:  PRINT=TABLES, DEPVAR=(V35,98), ITER=100, CONV=(V4-V8)
   ```

   DEPVAR=(variable number, maxcode)
   Variable number and maximum code for the dependent variable.
   No default; the variable number must always be specified.
   Default for maxcode is 9999999.

   CONVARS=(variable list)
   Variables to be used as predictors. If only one variable is given, a one-way analysis of variance will be performed.
   No default.

   MDVALUES=**BOTH**/MD1/MD2/NONE
   Which missing data values for the dependent variable are to be used. See "The IDAMS Setup File" chapter.
   Note: Missing data values are never checked for predictor variables.

   WEIGHT=variable number
   The weight variable number if the data are to be weighted.

ITERATIONS=**25**/n
The maximum number of iterations. Range: 1-99999.

TEST=**PCTMEAN**/CUTOFF/PCTRATIO/NONE
The convergence test desired.

PCTM	Test whether the change in all coefficients from one iteration to the next is below a specified fraction of the grand mean.
CUTO	Test whether the change in all coefficients from one iteration to the next is less than a specified value.
PCTR	Test whether the change in all coefficients from one iteration to the next is less than a specified fraction of the ratio of the standard deviation of the dependent variable to its mean.
NONE	The program will iterate until the maximum number of iterations has been exceeded.

CRITERION=**.005**/n
Supply a numeric value which is the tolerance of the convergence test selected. It ranges from 0.0 to 1.0. (Enter the decimal point).

OUTLIERS=**INCLUDE**/EXCLUDE

INCL	Cases with outlying values of the dependent variable will be counted and included in the analysis.
EXCL	Outliers will be excluded from the analysis.

OUTDISTANCE=**5**/n
Number of standard deviations from its grand mean used to define an outlier for the dependent variable.

WRITE=RESIDUALS
Write residuals to an IDAMS dataset; apply the MCA model only to the subset of cases passing missing data, maximum-code, and outlier criteria. Cases to which the MCA model does not apply are included in the residuals dataset with all values (except the identifying variable value) set to MD1.
Residuals cannot be obtained if only one predictor variable is specified.

OUTFILE=**OUT**/yyyy
A 1-4 character ddname suffix for the residuals output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.
Note: If more than one analysis requests residual output, the default ddnames DICTOUT and DATAOUT can only be used for one.

IDVAR=variable number
Number of an identification variable to be included in the residuals dataset.
Default: A variable is created whose values are numbers indicating the sequential position of the case in the residuals file.

PRINT=(TABLES, HISTORY, RESIDUALS)

TABL	Print the pair-wise cross-tabulations of the predictors.
HIST	Print the coefficients from all iterations. If the HIST option is not selected and if the iterations converge, only the final coefficients are printed; if the iterations do not converge, the coefficients from only the last 2 iterations are printed.
RESI	Print residuals in input case sequence order.

29.8 Restrictions

1. The maximum number of input variables, including variables used in Recode statements is 200.

2. Maximum number of predictor (control) variables per analysis is 50.
3. It is not possible to use the maximum number of predictors, each with the maximum number of categories, in an analysis. If a problem exceeds the available memory, an error message is printed, and the program skips to the next analysis.
4. Maximum number of analyses per execution is 50.
5. Predictor variables for multiple classification analysis must be categorized, preferably with 6 or fewer categories. The categories must have integer codes in the range 0-31. Cases with any other value will be dropped from the analysis.
6. Predictor variable for one-way analysis of variance must be coded in the range 0-2999. Cases with any other value are dropped from the analysis.
7. If a predictor variable has decimal places, only the integer part is used.
8. If the ID variable is alphabetic with width > 4, only the first four characters are used.

29.9 Examples

Example 1. Multiple classification analysis using four control variables (predictors): V7, V9, V12, V13, and dependent variable V100; separate analyses will be performed on the whole dataset and on two subsets of cases.

```
$RUN MCA
$FILES
PRINT  = MCA1.LST
DICTIN = LAB.DIC                        input Dictionary file
DATAIN = LAB.DAT                        input Data file
$SETUP
ALL RESPONDENTS TOGETHER
*                                (default values taken for all parameters)
DEPV=V100  CONV=(V7,V9,V12-V13)
$RUN MCA
$SETUP
INCLUDE V4=21,31-39
ONLY SCIENTISTS
*                                (default values taken for all parameters)
DEPV=V100  CONV=(V7,V9,V12-V13)
$RUN MCA
$SETUP
INCLUDE V4=41-49
ONLY TECHNICIANS
*                                (default values taken for all parameters)
DEPV=V100  CONV=(V7,V9,V12-V13)
```

Example 2. Multiple classification analysis with dependent variable V201 and three predictor variables V101, V102, V107; data are to be weighted by variable V6; producing residuals dataset where cases are identified by variable V2; cases with extreme values (outliers of more than 4 standard deviations from THE GRAND mean) on dependent variable are to be excluded from analysis. Residuals for the 1st 20 cases are listed afterwards using the LIST program.

```
$RUN MCA
$FILES
PRINT   = MCA2.LST
DICTIN  = LAB.DIC                        input Dictionary file
DATAIN  = LAB.DAT                        input Data file
DICTOUT = LABRES.DIC                     Dictionary file for residuals
DATAOUT = LABRES.DAT                     Data file for residuals
$SETUP
MULTIPLE CLASSIFICATION ANALYSIS  -  RESIDUALS WRITTEN INTO A FILE
*                                (default values taken for all parameters)
DEPV=V201  OUTL=EXCL  OUTD=4  IDVA=V2  WRITE=RESI  -
   CONV=(V101,V102,V107)  WEIGHT=V6
$RUN LIST
$SETUP
LISTING START OF RESIDUAL FILE
MAXCASES=20  INFILE=OUT
```

Example 3. For a dependent variable V52, interactions between three variables (V7, V9, V12) will be checked. V7 is coded 1,2,9, V9 is coded 1,3,5,9 and V12 is coded 0,1,9 where 9's are missing values. A single combination variable is constructed using Recode. This involves recoding each variable to a set of contiguous codes starting from zero and then using the COMBINE function to produce a unique code for each possible combination of codes for the three separate variables. MCA is performed using the 3 separate variables as predictors and a one-way analysis of variance is performed using the combination variable as control. Cases with missing data on the predictors will be excluded. Cases with values greater than 90000 on the dependent variable will also be excluded.

```
$RUN MCA
$FILES
DICTIN = CON.DIC                         input Dictionary file
DATAIN = CON.DAT                         input Data file
$SETUP
EXCLUDE V7=9 OR V9=9 OR V12=9
CHECKING INTERACTIONS
BADD=SKIP
DEPV=(V52,90000)  CONVARS=(V7,V9,V12)
DEPV=(V52,90000)  CONVARS=R1
$RECODE
   R7=V7-1
   R9=BRAC(V9,1=0,3=1,5=2)
   R1=COMBINE R7(2),R9(3),V12(2)
```

Chapter 30

Multivariate Analysis of Variance (MANOVA)

30.1 General Description

MANOVA performs univariate and multivariate analysis of variance and of covariance, using a general linear model. Up to eight factors (independent variables) can be used. If more than one dependent variable is specified, both univariate and multivariate analyses are performed. The program accepts both equal and unequal numbers of cases in the cells.

MANOVA is the only IDAMS program for multivariate analysis of variance. ONEWAY is recommended for one-way univariate analysis of variance. MCA handles multifactor univariate problems. It has no limitations with respect to empty cells, accepts more than 8 predictors, and allows for more than 80 cells. However, the basic analytic model of MCA is different from that of MANOVA. One important difference is that MCA is insensitive to interaction effects.

Hierarchical regression model. MANOVA uses a regression approach to analysis of variance. More particularly, the program employs a hierarchical model. There is an important consequence for the user: if a MANOVA execution involves more than 1 factor variable, and if there are disproportionate number of cases in the cells formed by the cross-classification of the factors, then consideration must be given to the order in which factor variables are specified. Disproportionality of subclass numbers confounds the main effects and the researcher must choose the order in which the confounded effects should be eliminated. When using MANOVA, this choice is accomplished by the order in which factor variables are specified. When using standard ordering, variables early in the specification have the effects of later variables removed, e.g. the first listed effect will be tested with all other main effects eliminated. The general rule is that each test eliminates effects listed before it on the test name specifications and ignores effects listed afterward. For a standard two-way analysis, the interaction term is not affected by the order of factor variables; more generally, for a standard n-way analysis, the n-th order interaction term and that term only, is unaffected. The problem exists for both univariate and multivariate analysis.

Contrast option. Two options are available for setting up contrasts (see the factor parameter CONTRAST). Nominal contrasts are generated by default; they are the customary deviations of row and column means from the grand mean and the generalization of these for the interaction contrasts. The program can also generate Helmert contrasts.

Augmentation of within cells sum of squares. It is possible to augment the within cells sum of squares (error term) using the orthogonal estimates (see the parameter AUGMENT). This allows the program to be used for Latin squares and for pooling of interaction terms with error.

Reordering and/or pooling orthogonal estimates. A conventional ordering of orthogonal estimates of effects (e.g. mean, C, B, A, BxC, AxC, AxB, AxBxC for three-factor design) is build into the program for standard usage. However, orthogonal estimates may be rearranged into some other order (see the parameter REORDER). Further, it is possible to pool several orthogonal estimates, such as several interaction terms, for simultaneous testing or to partition the cluster of orthogonal estimates for a given effect into smaller clusters for separate testing (see the test name parameter DEGFR).

30.2 Standard IDAMS Features

Case and variable selection. The standard filter is available for selecting cases for the execution. Dependent variables are selected by the parameter DEPVARS and covariates by the parameter COVARS. Factor variables are specified on special factor statements.

Transforming data. Recode statements may be used. Note that only integer values (positive or negative) are accepted for variables used as factors.

Weighting data. Use of weight variables is not applicable.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases with missing data codes on any of the input variables (dependent, covariate or factor variables) are excluded. This may result in many excluded cases and constitutes a potential problem which should be considered when planning an analysis.

30.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Cell means and N's. For each cell, N is printed and the mean for each dependent variable and covariate. The means are not adjusted for any covariates. Cells are labelled consecutively starting with "1 1" (for a 2 factor design) regardless of actual codes of factor variables. In indexing the cells, the indices of the last factor are the minor indices (fastest moving).

Basis of design. This is the design matrix generated by the program. The effects equations are in columns beginning with the mean effect in column 1. If REORDER was specified, the matrix is printed after reordering.

Intercorrelations among the coefficients of the normal equations.

Error correlation matrix. In a multivariate analysis of variance, the error term is a variance-covariance matrix. This is that error term (before adjustment for covariates, if any) reduced to a correlation matrix.

Principal components of the error correlation matrix. The components are in columns. These are the components of the error term (before adjustment for covariates, if any) of the analysis.

Error dispersion matrix and the standard errors of estimation. This is the error term, a variance-covariance matrix, for the analysis. The matrix is adjusted for covariates, if any. Each diagonal element of the matrix is exactly what would appear in a conventional analysis of variance table as the within mean square error for the variable. Degrees of freedom are adjusted for augmentation if that was requested. Standard errors of estimation correspond to the square roots of the diagonal elements of the matrix.

For analysis with covariate(s)

Adjusted error dispersion matrix reduced to correlations. This is the error term, a variance-covariance matrix, after adjustments for covariates, reduced to a correlation matrix.

Summary of regression analysis.

Principal components of the error correlation matrix after covariate adjustments. The components are in columns. These are the components of the error term of the analysis after adjustment for covariates.

For univariate analysis

An anova table. Degrees of freedom, sum of squares, mean squares and F-ratios.

For multivariate analysis

The following items are printed for each effect. Adjustments are made for covariates, if any. The order of effects is exactly opposite to the order of the test name specifications.

F-ratio for the likelihood ratio criterion. Rao's approximation is used. This is a multivariate test of

significance of the overall effect for all the dependent variables simultaneously.

Canonical variances of the principal components of the hypothesis. These are the roots, or eigenvalues, of the hypothesis matrix.

Coefficients of the principal components of the hypothesis. These are the correlations between the variables and the components of the hypothesis matrix. The number of nonzero components for any effect will be the minimum of the degrees of freedom and the number of dependent variables.

Contrast component scores for estimated effects. These are the scores of the hypothesis for the contrasts used in the design. They are analogous to the column means in a univariate analysis of variance and can be used in the same manner to locate variables and contrasts which give unusual departures from the null hypothesis.

Cumulative Bartlett's tests on the roots. This is an approximate test for the remaining roots after eliminating the first, second, third, etc.

F-ratios for univariate tests. These are exactly the F-ratios which would be obtained in a conventional univariate analysis.

30.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. All variables must be numeric. The dependent variable(s) and covariate(s) should be measured on an interval scale or be a dichotomy. The factor variables may be nominal, ordinal or interval but must have integer values; they are used to designate the proper cell for the case.

30.5 Setup Structure

```
$RUN MANOVA

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Factor specifications
        (repeated as required; at least one must be provided)
     5. Test name specifications
        (repeated as required; at least one must be provided)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

30.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further description of the program control statements, items 1-5 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  INCLUDE V2=1-4 AND V15=2
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  ANALYSIS OF AGE AND SALARY WITH SEX AND PROFESSION AS FACTORS
```

3. **Parameters** (mandatory). For selecting program options.

```
Example: DEPVARS=(V5,V8)  COVA=(V101,V102)
```

INFILE=**IN**/xxxx
: A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
: Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
: The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
: Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

DEPVARS=(variable list)
: A list of variables to be used as dependent variables.
No default.

COVARS=(variable list)
: A list of variables to be used as covariates.

AUGMENT=(m,n)
: To form error term, within sum of squares will be augmented by the columns m,m+1,m+2,...,n of the orthogonal estimates matrix.
Default: Within sum of squares will be used as the error term.

REORDER=(list of values)
: Reorder the orthogonal estimates according to the list (see the paragraph "Reordering and/or pooling orthogonal estimates" above). Note that if reordering of estimates is requested, the order of the test name specifications should correspond to the new order.
Example: the conventional ordering for a three-factor design can be changed to the order: mean, A, B, C, AxB, AxC, BxC, AxBxC using REORDER=(1,4,3,2,7,6,5,8).

PRINT=CDICT/DICT
: CDIC Print the input dictionary for the variables accessed with C-records if any.
DICT Print the input dictionary without C-records.

4. **Factor specifications** (at least one must be provided). Up to 8 factor specifications may be supplied. The coding rules are the same as for parameters. Each factor specification must begin on a new line.

```
Example:  FACTOR=(V3,1,2)
```

FACTOR=(variable number, list of code values)
Variable to be used as factor, followed by the code values which should be used to designate proper cell to the case.

CONTRAST=**NOMINAL**/HELMERT
Specifies the type of contrast to be used in computation.

NOMI	Nominal contrasts. Effect means deviated from the grand mean, i.e. M(1)-GM, M(2)-GM, etc.
HELM	Helmert contrasts. Mean of effect 1 deviated from the sum of means 1 through r, where r levels are involved.

5. **Test name specifications** (at least one must be provided). These specifications identify the tests that should be performed. They must be in the correct order. Ordinarily, there will be a specification for the grand mean, followed by a name specification for each main effect, and finally, a name specification for each possible interaction. If the design parameters are reordered or the degrees of freedom are regrouped (see the parameters REORDER and DEGFR), the test name statements must be made to conform to the modifications. The coding rules are the same as for parameters. Each test name specification must begin on a new line.

```
Example:  TESTNAME='grand mean'
```

TESTNAME='test name'
Up to 12 character name for each test to be performed. Primes are mandatory if the name contains non-alphanumeric characters.

DEGFR=n
The natural grouping of degrees of freedom (or hypothesis parameter equations) occures when the conventional ordering of statistical tests is used. DEGFR is used only to change the grouping, e.g. when you want to pool several interaction terms and test them simultaneously or to partition the degrees of freedom of some effect into two or more parts. When using the DEGFR parameter, be sure to use it on all test name statements, including a degree of freedom for the grand mean. Default: Use the natural grouping of degrees of freedom.

30.7 Restrictions

1. The maximum number of dependent variables is 19.
2. The maximum number of covariates is 20.
3. The maximum number of factor specifications is 8.
4. The maximum number of code values on a factor specification is 10.
5. The maximum number of cells is 80.
6. Cells with zero frequencies, with only one case, or with multiple identical cases, sometimes cause problems; the execution may end prematurely, or it may go to the end but produce invalid F-ratios and other statistics.

30.8 Examples

Example 1. Univariate analysis of variance (V10 is the dependent variable) with two factors represented by A with codes 1,2,3 and B with codes 21 and 31; nominal contrasts will be used in calculations, and tests will be performed in a conventional order.

```
$RUN MANOVA
$FILES
PRINT   = MANOVA1.LST
DICTIN  = CM-NEW.DIC                    input Dictionary file
DATAIN  = CM-NEW.DAT                    input Data file
$SETUP
UNIVARIATE ANALYSIS OF VARIANCE
DEPVARS=v10
FACTOR=(V3,1,2,3)
FACTOR=(V8,21,31)
TESTNAME='grand mean'
TESTNAME=B
TESTNAME=A
TESTNAME=AB
```

Example 2. Multivariate analysis of variance (V11-V14 are dependent variables) with two factors ("sex" coded 1,2 and "age" coded 1,2,3); nominal contrasts will be used in calculations, and tests will be performed in a conventional order.

```
$RUN MANOVA
$FILES
     as for Example 1
$SETUP
MULTIVARIATE ANALYSIS OF VARIANCE
DEPVARS=(v11-v14)
FACTOR=(V2,1,2)
FACTOR=(V5,1,2,3)
TESTNAME='grand mean'
TESTNAME=age
TESTNAME=sex
TESTNAME='sex & age'
```

Example 3. Multivariate analysis of variance (V11-V14 are dependent variables) with three factors (A coded 1,2, B coded 1,2,3, C coded 1,2,3,4); nominal contrasts will be used in calculations, and tests will be performed in a modified order (mean, A, B, AxB, C, AxC, BxC, AxBxC).

```
$RUN MANOVA
$FILES
     as for Example 1
$SETUP
MULTIVARIATE ANALYSIS OF VARIANCE - TESTS IN MODIFIED ORDER
DEPVARS=(v11-v14) REORDER=(1,4,3,7,2,6,5,8)
FACTOR=(V2,1,2)
FACTOR=(V5,1,2,3)
FACTOR=(V8,1,2,3,4)
TESTNAME=mean
TESTNAME=A
TESTNAME=B
TESTNAME=AxB
TESTNAME=C
TESTNAME=AxC
TESTNAME=BxC
TESTNAME=AxBxC
```

Chapter 31

One-Way Analysis of Variance (ONEWAY)

31.1 General Description

ONEWAY is a one-way analysis of variance program. An unlimited number of tables, using various independent and dependent variable pairs, may be produced in a single execution. Each analysis may be performed on all the cases or on a subset of cases of the data file; the selection of cases for one analysis is independent of the selection for other analyses. The term "control variable" used in ONEWAY is equivalent to "independent variable", "predictor" or, in analysis of variance terminology, "treatment variable".

An alternative to ONEWAY is the MCA program when only one predictor is specified. It permits a maximum code of 2999 for a control variable, whereas ONEWAY is limited to a maximum code of 99.

31.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. This filter affects all analyses in an execution. In addition, up to two local filters are available for independently selecting a subset of the data cases for each analysis. If two local filters are used, a case must satisfy both of them in order to be included in the analysis. Variables are selected for each analysis by the table parameters DEPVARS and CONVARS. A separate table is produced for each variable from the DEPVARS list with each variable from the CONVARS list.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES table parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases with missing data on the dependent variable are always excluded. Cases with missing data on the control variable may be optionally excluded (see the table parameter MDHANDLING).

31.3 Results

Table specifications. A list of table specifications providing a table of contents for the results.

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Descriptive statistics within categories of the control variable. Intermediate statistics are printed in table form for each code value of the control variable showing:

the number of valid cases (N) and sum of weights (rounded to nearest integer),
sum of weights as percent of the total sum,
mean, standard deviation, coefficient of variation, sum and sum of squares of dependent variable,
sum of dependent variable as percent of the total sum.

A totals row is printed for the table giving sums over all categories of the control variable (except categories with zero degrees of freedom, which are excluded from totals).

Analysis of variance statistics. Categories of the control variable which have zero degrees of freedom are not included in the computation of these statistics. The following statistics are printed for each table:

total sum of squares of the dependent variable,
eta and eta squared (unadjusted and adjusted),
the sum of squares between groups (between means sum of squares) and sum of squares within groups,
the F-ratio (printed only if the data are unweighted).

31.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. All analysis variables must be numeric; they may be integer or decimal valued.

A dependent variable should be measured on an interval scale or be a dichotomy. A control variable may be nominal, ordinal or interval but must have values in the range 0-99. If, for any case, the control variable for an analysis has a value exceeding this range, the case is eliminated from that analysis; no message is given. If the value of the control variable has decimal places, only the integer part is used (e.g. 1.1 and 1.6 are both placed in group 1); no message is given.

31.5 Setup Structure

```
$RUN ONEWAY

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Table specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

31.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of the cases to be used in the execution.

```
Example:  EXCLUDE V3=9
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  DATA ON TRAINING EFFECTS FOR FOOTBALL PLAYERS
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  *
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

PRINT=CDICT/DICT
CDIC Print the input dictionary for the variables accessed with C-records if any.
DICT Print the input dictionary without C-records.

4. **Table specifications.** The coding rules are the same as for parameters. Each table specification must begin on a new line.

```
Examples:  CONV=V6 DEPV=V26 WEIG=V3 F1=(V14,2,7) F2=(V13,1,1)
           CONV=V5 DEPV=(V27-V29,V80)
```

DEPVARS=(variable list)
A list of variables to be used as dependent variables

CONVARS=(variable list)
A list of variables to be used as control variables.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this set of tables. See "The IDAMS Setup File" chapter.

MDHANDLING=**DELETE**/KEEP
DELE Delete cases with missing data on the control variable.
KEEP Include cases with missing data on the control variable.
Note: Cases with missing data on the dependent variable are always deleted.

F1=(variable number, minimum valid code, maximum valid code)
F1 refers to the first filter variable which is used to create a subset of the data. The variable number should be the number of the filter variable; cases whose values for this variable fall in the minimum-maximum range will be entered in the table. The minimum value may be a negative integer. The maximum must be less than 99,999. Decimal places must be entered where appropriate.

F2=(variable number, minimum valid code, maximum valid code)
F2 refers to the second filter variable. If this second filter is specified, a case must satisfy the requirements of both filters to enter the table.

31.7 Restrictions

1. The maximum number of control variables is 99. The maximum number of dependent variables is 99. The total number of variables which may be accessed is 204, including variables used in Recode statements.
2. ONEWAY uses control variable values in the range 0 to 99. If, for any case, the control variable for a certain analysis has a value exceeding this range, the case is eliminated from that table.
3. The maximum sum of weights is about 2,000,000,000.
4. The F-ratio is printed for unweighted data only.

31.8 Examples

Example 1. Three one-way analyses of variance using V201 as control and V204 as dependent variable: first for the whole dataset, second for a subset of cases having values 1-3 for variable V5, and the third for a subset of cases having values 4-7 for variable V5.

```
$RUN ONEWAY
$FILES
PRINT  = ONEW1.LST
DICTIN = STUDY.DIC                      input Dictionary file
DATAIN = STUDY.DAT                      input Data file
$SETUP
ONE-WAY ANALYSES OF VARIANCE DESCRIBED SEPARATELY
*                       (default values taken for all parameters)
CONV=V201  DEPV=V204
CONV=V201  DEPV=V204  F1=(V5,1,3)
CONV=V201  DEPV=V204  F1=(V5,4,7)
```

Example 2. Generation of a one-way analysis of variance for all combinations of control variables V101, V102, V105 and V110, and dependent variables V17 through V21; data are weighted by variable V3.

```
$RUN ONEWAY
$FILES
    as for Example 1
$SETUP
MASS-GENERATION OF ONE-WAY ANALYSES OF VARIANCE
*                       (default values taken for all parameters)
CONV=(V101,V102,V105,V110)  DEPV=(V17-V21)  WEIGHT=V3
```

Chapter 32

Partial Order Scoring (POSCOR)

32.1 General Description

POSCOR calculates (ordinal scale) scores using a procedure based on the hierarchical position of the elements in a partially ordered set according to a number of properties (or characteristics, etc.). The scores, calculated separately for each element of the set, are output to a Data file described by an IDAMS dictionary. This file can then be used as input to other analysis programs.

Using the ORDER parameter, different types of scores can be obtained, namely: (1) four types of scores where calculations are based on the proportion of cases dominated by the case; (2) four other scores where calculations are based on the proportion of cases which dominate the case examined. The range of the scores is determined by the SCALE parameter. Meaningful score values can be expected only when the number of cases involved is much greater than the number of variables (or components of the score) specified.

In applications with variables of not uniform importance, a priority list can be defined using the analysis parameter LEVEL in the partial ordering. If the variables of higher priority unambiguously determine the relation of two cases, the variables of lower priority are not considered.

In the special case when only one variable is used in an analysis, the transformed values correspond to their probabilities (see ORDER=ASEA/DEEA/ASCA/DESA options).

In one analysis, a series of mutually exclusive subsets can be examined using the subset facility. In this event, the score variable(s) are computed within each subset of cases.

32.2 Standard IDAMS Features

Case and variable selection. The standard filter is available for selecting cases for the execution. A case subsetting option is also available for each analysis. Variables to be transferred to the output file are selected using the TRANSVARS parameter. Variables for each analysis are selected in the analysis specifications.

Transforming data. Recode statements may be used. Note that only integer part of recoded variables is used by the program, i.e. recoded variables are rounded to the nearest integer.

Weighting data. Use of weight variables is not applicable.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. The MDHANDLING parameter indicates whether variables or cases with missing data are to be excluded from an analysis.

32.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Output dictionary. (Optional: see the parameter PRINT).

32.4 Output Dataset

The output file contains the computed scores along with transferred variables and, optionally, analysis variables, for each case used in the analysis (i.e. all cases passing the filter and not excluded through the use of the missing data handling option). An associated IDAMS dictionary is also output.

Output variables are numbered sequentially starting from 1 and have the following characteristics:

- *Analysis and subset variables* (optional: only if AUTR=YES). V-variables have the same characteristics as their input equivalents. Recode variables are output with WIDTH=7 and DEC=0.
- *Case identification (ID) and transferred variables.* V-variables have the same characteristics as their input equivalents. Recode variables are output with WIDTH=7 and DEC=0.
- *Computed score variables.*

  For ORDER=ASEA/DEEA/ASCA/DESA, one variable for each analysis with:

Name	specified by ANAME	(default: blank)
Field width	specified by FSIZE	(default: 5)
No. of decimals	0	
MD1	specified by OMD1	(default: 99999)
MD2	specified by OMD2	(default: 99999)

  For ORDER=ASER/DESR/ASCR/DEER, two variables for each analysis with names specified by ANAME and DNAME parameters respectively and other characteristics as outlined above.

Note. If an analysis is repeated for several mutually exclusive subsets of cases, the score variable is computed for the cases in each subset in turn. If a case does not fall into any of the defined subsets for the analysis, then its score variable(s) values will be set to the MD1 code.

32.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. For analysis variables, only integer values are used. Decimal values, if any, are rounded to the nearest integer. The case ID variable and variables to be transferred can be alphabetic.

32.6 Setup Structure

```
$RUN POSCOR

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Subset specifications (optional)
     5. POSCOR
     6. Analysis specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary
DATAyyyy   output data
PRINT      results (default  IDAMS.LST)
```

32.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further description of the program control statements, items 1-3 and 6 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V2=1-4 AND V15=2
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  SCALING THE RU INPUT VARIABLES
   ```

3. **Parameters** (mandatory). For selecting program options.

   ```
   Example: MDHAND=CASES  TRAN=V5 IDVAR=R6
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=**VARS**/CASES
Treatment of missing data.

VARS	A variable containing a missing data value is excluded from the comparison.
CASE	A case containing a missing data value is excluded from the analysis.

OUTFILE=**OUT**/yyyy
A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

IDVAR=variable number
Variable to be transferred to the output dataset to identify the cases.
No default.

TRANSVARS=(variable list)
Additional variables (up to 99) to be transferred to the output dataset. This list should not include analysis variables or variables used in subset specifications. These are transferred automatically using the AUTR parameter.

AUTR=**YES**/NO

YES	Analysis variables and variables used in subset specifications will be automatically transferred to the output dataset.
NO	No transfer of analysis and subset variables.

FSIZE=**5**/n
Field width of the variables (scores) computed.

SCALE=**100**/n
The value (scale factor) specifying the range (0 - n) of the scores computed.

OMD1=**99999**/n
Value of the first missing data code for the computed variables (scores).

OMD2=**99999**/n
Value of the second missing data code for the computed variables (scores).

PRINT=(CDICT/DICT, **OUTDICT**/OUTCDICT/NOOUTDICT)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
OUTD	Print the output dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
NOOU	Do not print the output dictionary.

4. **Subset specifications** (optional). These specify mutually exclusive subsets of cases for a particular analysis.

```
Example:  AGE      INCLUDE V5=15-20,21-45,46-64
```

Rules for coding

Prototype: name statement

name
Subset name. 1-8 alphanumeric characters beginning with a letter. This name must match exactly the name used on subsequent analysis specifications. Embedded blanks are not allowed. It is recommended that all names be left-justified.

statement
Subset definition.

- Start with word INCLUDE.
- Specify variable number (V- or R-variable) on which subsets are to be based (alphabetic variables are not allowed).
- Specify values and/or ranges of values separated by commas. Each value or range defines one subset. Commas separate the subsets. Negative ranges must be expressed in numeric sequence, e.g. -4 - -2 (for -4 to -2); -2 - 5 (for -2 to +5). The subsets must be mutually exclusive (i.e. same values cannot appear in two ranges). In the example above, 3 subsets based on the value of V5 are defined for the AGE subset specification.
- Enter a dash at the end of one line to continue to another.

5. **POSCOR.** The word POSCOR on this line signals that analysis specifications follow. It must be included (in order to separate subset specifications from analysis specifications) and must appear only once.

6. **Analysis specifications.** The coding rules are the same as for parameters. Each analysis specification must begin on a new line.

```
Example:  ORDER=ASER  ANAME=MSDCORE  DNAME=DOWNSCORE  -
              VARS=(V3-V6)  LEVELS=(1,1,2,2)
```

VARS=(variable list)
The V- and/or R-variables to be used in the analysis.
No default.

ORDER=**ASEA**/DEEA/ASCA/DESA/ASER/DESR/ASCR/DEER
Specifies the type of score to be computed.
The score is based upon:

ASEA	cases better or equal/dominating
DEEA	cases worse or equal/dominated
ASCA	cases strictly better/strictly dominating
DESA	cases strictly worse/strictly dominated
	relatively to the **total** number of cases

ASER/DESR

ASER	cases better or equal/dominating
DESR	cases strictly worse/strictly dominated
	relatively to the number of **comparable** cases

ASCR/DEER

ASCR	cases strictly better/strictly dominating
DEER	cases worse or equal/dominated
	relatively to the number of **comparable** cases

Note. In both latter cases the two scores are computed whatever is selected. The sum of them equals the value specified in the SCALE parameter.

SUBSET=xxxxxxxx
: Specifies the name of the subset specification to be used, if any. Enclose the name in primes if it contains non-alphanumeric characters. Upper case letters should be used in order to match the name on the subset specification which is automatically converted to upper case.

LEVELS=**(1, 1,..., 1)** / (N1, N2, N3,...,Nk)
: "k" is the number of variables used in the analysis variable list. Ni defines the priority order of the i-th variable in the list of variables involved in the partial ordering. A higher value implies a lower priority. The priority values must be specified in the same sequence as the corresponding variables in the analysis variable list. The default of all 1's implies that all variables have the same priority.

ANAME='name'
: Up to 24 character name for the increasing score. Primes are mandatory if the name contains non-alphanumeric characters.
Default: Blanks.

DNAME='name'
: Up to 24 character name for the decreasing score. Primes are mandatory if the name contains non-alphanumeric characters.
Default: Blanks.

32.8 Restrictions

1. The values of the analysis variables must be between -32,767 and +32,767.
2. Components of the priority list in the LEVEL parameter must be positive integers between 1 and 32,767.
3. Maximum number of analyses is 10.
4. Maximum number of variables to be transferred is 99.
5. A variable can only be used once whether it be an ID variable, in an analysis list or in a transfer list. If it is required to use the same variable twice, then use recoding to obtain a copy with a different variable (result) number.
6. Maximum number of variables used for analysis, in subset specifications and in a transfer list is 100 (including both V- and R-variables).
7. Maximum number of subset specifications is 10.
8. If the ID variable or a variable to be transferred is alphabetic with width > 4, only the first four characters are used.
9. Although the number of cases processed is not limited, it should be noted that the execution time increases as a quadratic function of the number of cases being analysed.

32.9 Examples

Example 1. Computation of two scores using the same variables V10, V12, V35 through V40; the first score will be calculated on the whole dataset, while the second one will be calculated separately on three subsets (for values 1, 2 and 3 of the variable V7); cases with missing data are to be excluded from analyses; both scores are based upon the cases strictly dominated relative to the number of comparable cases; cases are identified by variables V2 and V4 which are transferred to the output file. Note that Recode is used to make a copy of the variables since a restriction of the program means that a variable may only be used once in an execution.

```
$RUN POSCOR
$FILES
PRINT   = POSCOR1.LST
DICTIN  = PREF.DIC                      input Dictionary file
DATAIN  = PREF.DAT                      input Data file
DICTOUT = SCORES.DIC                    output Dictionary file
DATAOUT = SCORES.DAT                    output Data file
$SETUP
COMPUTATION OF TWO SCORES
MDHAND=CASES  IDVAR=V2  TRANSVARS=V4
TYPE     INCLUDE V7=1,2,3
POSCOR
ORDER=DESR  ANAME='GLOBAL SCORE  INCR'  DNAME='GLOBAL SCORE  DECR' -
   VARS=(V10,V12,V35-V40)
ORDER=DESR  ANAME='ADJUSTED SCORE   INCR' -
   DNAME='ADJUSTED SCORE   DECR'   SUBS=TYPE -
   VARS=(R10,R12,R35-R40)
$RECODE
   R10=V10
   R12=V12
   R35=V35
   R36=V36
   R37=V37
   R38=V38
   R39=V39
   R40=V40
```

Example 2. Computation of three scores based upon cases dominating relative to the total number of cases; analysis variables are not to be transferred to the output file; variables containing missing data values are to be excluded from the comparison; case identification variables V1 and V5 are transferred.

```
$RUN POSCOR
$FILES
     as for Example 1
$SETUP
COMPUTATION OF THREE SCORES
AUTR=NO  IDVAR=V1  TRANSVARS=V5
POSCOR
ORDER=ASEA  ANAME='SCORE 1   INCR'  VARS=(V11,V17,V55-V60)
ORDER=ASEA  ANAME='SCORE 2   INCR'  VARS=(V108-V110,V114,V116,V118,V120)
ORDER=ASEA  ANAME='SCORE 3   INCR'  VARS=(V22,V33,V101-V105)
```

Chapter 33

Pearsonian Correlation (PEARSON)

33.1 General Description

PEARSON computes and prints matrices of Pearson r correlation coefficients and covariances for all pairs of variables in a list (square matrix option) or for every pair of variables formed by taking one variable from each of two variable lists (rectangular matrix option).

Either "pair-wise" or "case-wise" deletion of missing data may be specified.

PEARSON can also be used to output a correlation matrix which can subsequently be input to the REGRESSN or MDSCAL programs. Although REGRESSN is capable of computing its own correlation matrix, its missing data handling is limited to "case-wise" deletion. In contrast, a matrix can be generated by PEARSON using a "pair-wise" deletion algorithm for missing data.

33.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. The variables for which correlations are desired are specified with the ROWVARS and COLVARS parameters.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. The univariate statistics for each variable are computed from the cases which have valid (non-missing) data for the variable.

Missing data: pair-wise deletion. Paired statistics and each correlation coefficient can be computed from the cases which have valid data for both variables (MDHANDLING=PAIR). Thus, a case may be used in the computations for some pairs of variables and not used for other pairs. This method of handling missing data is referred to as the "pair-wise" deletion algorithm. Note: If there are missing data, individual correlation coefficients may be computed on different subsets of the data. If there is a great deal of missing data, this can lead to internal inconsistencies in the correlation matrix which can cause difficulties in subsequent multivariate analysis.

Missing data: case-wise deletion. The program can also be instructed (MDHANDLING=CASE) to compute the paired statistics and correlations from the cases which have valid data on all variables in the variable list. Thus, a case is either used in computations for all pairs of variables or not used at all. This method of handling missing data is referred to as the "case-wise" deletion algorithm (also available in the REGRESSN program), and applies only to the square matrix option.

33.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Square matrix option

Paired statistics. (Optional: see the parameter PRINT). For each pair of variables in the variable list the following are printed:

number of valid cases (or weighted sum of cases),
mean and standard deviation of the X variable,
mean and standard deviation of the Y variable,
t-test for correlation coefficient,
correlation coefficient.

Univariate statistics. For each variable in the variable list the following are printed:

number of valid cases and sum of weights,
sum of scores and sum of scores squared,
mean and standard deviation.

Regression coefficients for raw scores. (Optional: see the parameter PRINT). For each pair of variables x and y, the regression coefficients a and c and the constant terms b and d in the regression equations x=ay+b and y=cx+d are printed.

Correlation matrix. (Optional: see the parameter PRINT). The lower-left triangle of the matrix.

Cross-products matrix. (Optional: see the parameter PRINT). The lower-left triangle of the matrix.

Covariance matrix. (Optional: see the parameter PRINT). The lower-left triangle of the matrix with diagonal.

In each of the above matrices, a maximum of 11 columns and 27 rows are printed per page.

Rectangular matrix option

Table of variable frequencies. Number of valid cases for each pair of variables.

Table of mean values for column variables. Means are calculated and printed for each column variable over the cases which are valid for each row variable in turn.

Table of standard deviations for column variables. As for means.

Correlation matrix. (Optional: see the parameter PRINT). Correlation coefficients for all pairs of variables.

Covariance matrix. (Optional: see the parameter PRINT). Covariances for all pairs of variables.

In each of the above tables, a maximum of 8 columns and 50 rows are printed per page.

Note: If a variable pair has no valid cases, 0.0 is printed for the mean, standard deviation, correlation and covariance.

33.4 Output Matrices

Correlation matrix

The correlation matrix in the form of an IDAMS square matrix is output when the parameter WRITE=CORR is specified. The format used to write the correlations is 8F9.6; the format for both the means and standard deviations is 5E14.7. Columns 73-80 are used to identify the records.

The matrix contains correlations, means, and standard deviations. The means and standard deviations are unpaired. The dictionary records which are output by PEARSON contain variable numbers and names from the input dictionary and/or Recode statements. The order of the variables is determined by the order of variables in the variable list.

PEARSON may generate correlations equal to 99.99901, and means and standard deviations equal to 0.0 when it is unable to compute a meaningful value. Typical reasons are that all cases were eliminated due to missing data or one of the variables was constant in value. Note that MDSCAL does not accept these "missing values" although REGRESSN does.

Covariance matrix

The covariance matrix without the diagonal in the form of an IDAMS square matrix is output when the parameter WRITE=COVA is specified.

33.5 Input Dataset

The input is a Data file described by an IDAMS dictionary. All analysis variables must be numeric; they may be integer or decimal valued.

33.6 Setup Structure

```
$RUN PEARSON

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
FT02       output matrices if WRITE parameter specified
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

33.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V2=11-15,60  OR  V3=9
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  FIRST EXECUTION OF PEARSON - APRIL 27
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  WRITE=CORR, PRINT=(CORR,COVA)  ROWV=(V1,V3-V6,R47,V25)
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MATRIX=**SQUARE**/RECTANGULAR
SQUA Compute Pearson correlation coefficients for all pairs of variables from the ROWV list.
RECT Compute Pearson correlation coefficients for every pair of variables formed by taking one variable from each of the ROWV and COLV lists.

ROWVARS=(variable list)
A list of V- and/or R-variables to be correlated (MATRIX=SQUARE) or the list of row variables (MATRIX=RECTANGULAR).
No default.

COLVARS=(variable list)
(MATRIX=RECTANGULAR only).
A list of V- and/or R-variables to be used as the column variables. Eight columns are printed per page; if either the row variable list or the column variable list contains less than eight variables, it is preferable (for ease of reading results) to have the short list as the column variable list.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=**PAIR**/CASE
Method of handling missing data.
PAIR Pair-wise deletion.
CASE Case-wise deletion (not available with MATRIX=RECTANGULAR).

WEIGHT=variable number
The weight variable number if the data are to be weighted.

WRITE=(CORR, COVA)
(MATRIX=SQUARE only).
CORR Output the correlation matrix with means and standard deviations.
COVA Output the covariance matrix with means and standard deviations.

PRINT=(CDICT/DICT, **CORR**/NOCORR, COVA, PAIR, REGR, XPRODUCTS)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
CORR	Print the correlation matrix.
COVA	Print the covariance matrix.
PAIR	Print the paired statistics (MATRIX=SQUARE only).
REGR	Print the regression coefficients (MATRIX=SQUARE only).
XPRO	Print the matrix of cross-products (MATRIX=SQUARE only).

33.8 Restrictions

When MATRIX=SQUARE is specified

1. The maximum number of variables permitted in an execution is 200. This limit includes all analysis variables, and variables used in Recode statements.
2. Recode variable numbers must not exceed 999 if the parameter WRITE is specified. (They are output as negative numbers in the descriptive part of the matrix which has only 4 columns reserved for the variable number e.g. R862 becomes -862).

When MATRIX=RECTANGULAR is specified

1. The maximum number of variables in either row or the column variable list is 100.
2. Maximum total number of row variables, column variables, variables used in Recode statements, and the weight variable is 136.

33.9 Examples

Example 1. Calculation of a square matrix of Pearson's r correlation coefficients with pair-wise deletion of cases having missing data; the matrix will be written into a file and printed.

```
$RUN PEARSON
$FILES
PRINT  = PEARS1.LST
FT02   = BIRDCOR.MAT                    output Matrix file
DICTIN = BIRD.DIC                       input Dictionary file
DATAIN = BIRD.DAT                       input Data file
$SETUP
MATRIX OF CORRELATION COEFFICIENTS
PRINT=(PAIR,REGR,CORR)  WRITE=CORR  ROWV=(V18-V21,V36,V55-V61)
```

Example 2. Calculation of Pearson's r correlation coefficients for variables V10-V20 with variables V5-V6.

```
$RUN PEARSON
$FILES
DICTIN = BIRD.DIC                       input Dictionary file
DATAIN = BIRD.DAT                       input Data file
$SETUP
CORRELATION COEFFICIENTS
MATRIX=RECT  ROWV=(V10-V20)  COLV=(V5-V6)
```

Chapter 34

Rank-Ordering of Alternatives (RANK)

34.1 General Description

RANK determines a reasonable rank-order of alternatives, using preference data as input and three different ranking procedures, one based on classical logic (the method ELECTRE) and two others based on fuzzy logic. The two approaches essentially differ in the way the relational matrices are constructed. With fuzzy ranking, the data completely determine the result whereas with classical ranking the user, relying on concepts of classical logic, has the possibility of controlling the calculation of the overall relations among alternatives.

The ELECTRE method (classical logic) implemented in RANK, in a first step, uses the input preference data to calculate a final matrix expressing the overall collective opinion about the "dominance" among alternatives, the structure of the relation not necessarily corresponding to a linear or partial order. The "dominance" relation for each pair of alternatives is controlled by the conditions for "concordance" and for "discordance" fixed by the user. Different relational structures may be obtained from the same data by varying the analysis parameters. In the second step, the procedure looks for a sequence of non-dominated layers (cores) of alternatives. The first core consists of the alternatives of highest rank in the whole set considered. It should be noted that in certain cases further cores may not exist due to loops in the relation. This may be true even at the highest level.

The first fuzzy method (non-dominated layers) was originally developed for solving decision-making problems with fuzzy information. This method makes it possible to find a sequence of non-dominated layers (cores) of alternatives in a fuzzy preference structure, which does not necessarily represent a (total) linear order. The subsequent cores are such groups of alternatives which have the highest rank among the alternatives which do not belong to previous, higher level cores. The first core stands for the alternatives of highest rank in the whole set considered.

The second fuzzy method (ranks) tries to find the credibility of the statements "the j-th alternative is exactly at the p-th position in the rank-order". The results are straight-forward in the case of a (total) linear order relation behind the data; otherwise special care should be given to the interpretation of the results. The optimization procedure, developed to handle the general (normalized or non-normalized) case, allows the user to decide whether to normalize the fuzzy relational matrix before the actual ranking procedure (see option NORM). A careful interpretation of the results is needed after normalization. Usually incomplete data result in a non-normalized relational matrix especially when DATA=RAWC is used and the number of selected alternatives in individual answers is smaller than the number of possible alternatives. Although a non-normalized matrix gives results in which the level of uncertainty is higher, it may provide a more realistic picture about the latent relation determining the data; indeed the normalization can be interpreted as a kind of extrapolation.

Two types of individual preference relations (strict or weak) can be specified, both in the case of data representing a selection of alternatives, and in the case of data representing a ranking of alternatives.

1. Data representing a **selection** of alternatives.
   - **Strict preference:** each selected alternative is considered to have a **unique** (different) rank, while the non-selected ones are given the same lowest rank.
   - **Weak preference:** all selected alternatives are considered to have same **common** rank, which is higher than the rank of the non-selected ones.
2. Data representing a **ranking** of alternatives.
   - **Strict preference:** all ranked alternatives are supposed to have **different** values, and relations between alternatives having the same rank are disregarded in the calculation of the overall preference relation across the alternatives.
   - **Weak preference:** alternatives with the **same** rank are taken into account in the calculation.

34.2 Standard IDAMS features

Case and variable selection. The standard filter is available to select a subset of cases from the input data, and the parameter VARS is used to select variables.

Transforming data. Recode statements may be used. Note that only integer part of recoded variables is used by the program, i.e. recoded variables are rounded to the nearest integer.

Weighting data. Data may be weighted by integer values. Note that decimal valued weights are rounded to the nearest integer. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. For DATA=RAWC, the variables with missing data are skipped; for DATA=RANKS, the missing data values are substituted by the lowest rank.

34.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Invalid data. Messages about incorrect (rejected) data.

Methods based on fuzzy logic (METHOD=NOND/RANKS)

Matrix of relations. A square matrix representing the fuzzy relation is printed by rows. If the rows have more than ten elements they are continued on subsequent line(s).

Description of the relations. After printing the type of relation, three measures are given which characterize concisely the relation, namely: absolute coherence, intensity and absolute dominance indices.

Analysis results. The results are presented in a different form for each method.

For METHOD=NOND the cores are printed sequentially from the highest rank and for each of them the following information is given:

its sequential number, with the certainty level,
the codes and code labels of the alternatives, or the variable numbers and names (up to 8 characters),
the membership function values of the alternatives indicating how strongly they are connected to the core; membership values of alternatives belonging to previous cores are substituted by asterisks,
list of alternatives belonging to the core with the highest membership value (most credible alternatives).

For METHOD=RANKS the normalized relational matrix is printed first if normalization was requested. The results are then printed, in two forms for easier interpretation.

1. All alternatives are listed sequentially with, for each:
   the code and code label of the alternative, or the variable number and name,
   the membership function values of the alternative indicating how strongly it is connected to each rank,
   the list of most credible rank(s) for that alternative.

2. All ranks are listed sequentially with, for each:
   the rank's number,
   the codes and code labels of the alternatives, or the variable numbers and names,
   the membership function values of the alternatives indicating how strongly they are connected to that rank,
   the list of most credible alternative(s) for that rank.

Method based on classical logic (METHOD=CLAS)

Analysis results. For each final "dominance" relational structure resulting from one analysis, the rank differences and the minimum/maximum population proportions specified by the user are printed, followed by the list of successive non-dominated cores (identified by their sequential number) with the alternatives belonging to them.

Note. Alternatives are labelled either with the first 8 characters of the variable label for DATA=RANKS or with the 8-character code label (if C-records are present in the dictionary) for DATA=RAWC.

34.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. All analysis variables must have positive integer values. Note that decimal valued variables are rounded to the nearest integer.

Preferences can be represented in 2 ways in the data. The following illustration shows these.

Suppose that data are to be collected about employee preferences for various factors relating to their job:

Own office
High salary
Long holidays
Minimum supervision
Compatible colleagues

The 2 ways of representing this in a questionnaire are:

1. DATA=RAWC
   In this case, the factors are coded (e.g. 1 to 5) and the respondent is asked to pick them in order of preference. The variables in the data would represent the rank, e.g.

```
V6  Most important factor
V7  2nd most important factor
.
.
V10 Least important factor
```

   and the codes assigned to each of these variables by a respondent would represent the factors (e.g. 1=own office, 2=high salary, etc.).

   Not all possible factors need be selected, one could ask say for the 3 most important, by specifying only these variables on the variable list e.g. V6, V7, V8. The number of different factors being used is specified with the NALT parameter.

2. DATA=RANKS
   Here, each factor is listed in the questionnaire as a variable, e.g.

```
V13 Own office
V14 High salary
.
.
V17 Compatible colleagues
```

and the respondent is invited to assign a rank to each, where 1 is given to the most important factor, 2 to the next most important, etc. Here the variables represent the factors and their values represent the rank. Each variable must be assigned a rank and all factors will always enter into the analysis. The ranks must be coded from 1 to n where n is the number of variables being considered.

Notes.

1. If DATA=RANKS, the code 0 and all codes greater than n where n is the number of variables (i.e. number of alternatives) are treated as missing values and are assigned to the lowest rank.
2. If DATA=RAWC, the first NALT different codes encountered while reading the data (excluding 0) are used as valid codes. Other codes encountered later in the data are taken as illegal codes. Zero is always treated as an illegal code. If the number of alternatives selected by the respondents is less than NALT, then the not selected alternatives appear on the results with zero code value and empty code label.

34.5 Setup Structure

```
$RUN RANK

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Analysis specifications (repeated as required)
        (for classical logic only)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

34.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further description of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

```
Example:  INCLUDE V2=11
```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example:  FIRST RUN OF RANK
```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  DATA=RANKS  PREF=STRICT MDVALUES=NONE  VARS=(V11-V13)
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.
For DATA=RAWC, variables with missing data are not included in the ranking.
For DATA=RANKS, missing data values are recoded to the lowest rank.

VARS=(variable list)
A list of V- and/or R-variables to be used in the ranking procedure.
No default.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

METHOD=(**CLASSICAL**/NOCLASSICAL, NONDOMINATED, RANKS)
Specifies the method to be used in the analysis.

CLAS	Method of classical logic (ELECTRE).
NOND	Fuzzy method-1, called non-dominated layers.
RANK	Fuzzy method-2, called ranks.

DATA=**RAWC**/RANKS
Type of data.

RAWC	The variables correspond to ranks (the first variable in the list has the first rank, the second one the second rank, etc.), while their value is the code number of the alternative selected.
RANK	Variables represent alternatives, their values being ranks of the corresponding alternatives.

PREF=**STRICT**/WEAK
Determines the type of the preference relation to be used in the analysis.
STRI A strict preference relation is used.
WEAK A weak preference relation is used.

NALT=**5**/n
(DATA=RAWC only). Total number of alternatives to be ranked.
Note: If DATA=RANKS, the number of alternatives is automatically set to the number of analysis variables.

NORMALIZE=**NO**/YES
(METHOD=RANKS only).
NO No normalization.
YES Normalization of the relational matrix is performed before calculating the value of membership function of alternatives.

PRINT=CDICT/DICT
CDIC Print the input dictionary for the variables accessed with C-records if any.
DICT Print the input dictionary without C-records.

4. **Analysis specifications** (conditional: only in case of classical logic method). The coding rules are the same as for parameters. Each analysis specification must begin on a new line.

```
Example:  PCON=66  DDIS=4   PDIS=20
```

DCON=**1**/n
Rank difference controlling the concordance in individual opinions (cases). It must be an integer in the range 0 to NALT-1.

PCON=**51**/n
Minimum proportion of individual concordance, expressed as a percentage, required in the collective opinion. It must be an integer in the range 0 to 99. The default value means that at least 51% agreement is requested for a collective concordance.

DDIS=**2**/n
Rank difference controlling the discordance in individual opinions (cases). It must be an integer in the range 0 to NALT-1.

PDIS=**10**/n
Maximum proportion of individual discordance, expressed as a percentage, tolerated in the collective opinion. It must be an integer in the range 0 to 100. The default value means that no more than 10% individual discordance is tolerated.

34.7 Restrictions

1. The maximum number of variables permitted in any execution is 200, including those used in Recode statements and the weight variable.
2. The maximum number of analysis variables is 60.

34.8 Examples

Example 1. Determination of a rank-order of alternatives using data collected in the form of ranking of alternatives; there are 10 alternatives, weak preference relation is assumed, and analysis is to be done using the Ranks method.

```
$RUN RANK
$FILES
PRINT  = RANK1.LST
DICTIN = PREF.DIC                    input Dictionary file
DATAIN = PREF.DAT                    input Data file
$SETUP
RANK - ORDERING OF ALTERNATIVES : RANKS METHOD
DATA=RANKS  PREF=WEAK  METH=(NOCL,RANKS)  VARS=(V21-V30)
```

Example 2. Determination of a rank-order of alternatives using data collected in the form of a selection of priorities; three alternatives are selected out of 20 and the order of variables determines the priority of selection; strict preference relation is assumed; both fuzzy methods are requested in analysis.

```
$RUN RANK
$FILES
    as for Example 1
$SETUP
RANK - ORDERING OF ALTERNATIVES : TWO FUZZY METHODS
NALT=20  METH=(NOCL,NOND,RANKS)  VARS=(V101-V103)
```

Example 3. Determination of a rank-order of alternatives using data collected in the form of a selection of priorities; 4 alternatives are selected out of 15 and the order of variables does not determine the priority of selection (weak preference); four classical logic analyses are to be performed keeping rank differences always equal to 1, but increasing proportion of discordance and decreasing proportion of concordance.

```
$RUN RANK
$FILES
    as for Example 1
$SETUP
RANK - ORDERING OF ALTERNATIVES : CLASSICAL LOGIC
PREF=WEAK  NALT=15  METH=CLAS  VARS=(V21,V23,V25,V27)
PCON=75  DDIS=1  PDIS=5
PCON=66  DDIS=1  PDIS=10
PCON=51  DDIS=1  PDIS=15
PCON=40  DDIS=1  PDIS=20
```

Chapter 35

Scatter Diagrams (SCAT)

35.1 General Description

SCAT is a bivariate analysis program which produces scatter diagrams, univariate statistics, and bivariate statistics. The scatter diagrams are plotted on a rectangular coordinate system; for each combination of coordinate values that appears in the data, the frequency of its occurrence is displayed.

SCAT is useful for displaying bivariate relationships if the numbers of different values for each variable is large and the number of data cases containing any one value is small. If, however, a variable assumes relatively few different values in a large number of data cases, the TABLES program is more appropriate.

Plot format. Each plot desired is defined separately by specifying the two variables to be used (called the X and Y variables). The scales of the axes are adjusted separately for each plot to allow variables with radically different scales to be plotted against each other without loss of discrimination. Normally, the program plots the variable with the greater range (before rescaling) along the horizontal axis. However, the user may request that the X variable always be plotted along the horizontal axis. The actual frequencies are entered into the diagram if they are less than 10. For frequencies from 10-65, the letters of the alphabet are used. If the frequency of a point is greater than 65, an asterisk is placed in the diagram. This coding scheme is part of the results for easy reference.

Statistics. The mean, standard deviation, minimum and maximum values are printed for each variable accessed, including the plot filter and weight variable, if any. For each plot the program also prints the mean, standard deviation, case count and range for the two variables, Pearson's correlation coefficient r, the regression constant, and the unstandardized regression coefficient for predicting Y from X.

35.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. In addition, a plot filter variable and range of values may be specified to restrict the data cases included in a particular plot. The variables to be plotted are specified in pairs with plot parameters.

Transforming data. Recode statements may be used. Note that for R-variables, the number of decimals to be retained is specyfied by the NDEC parameter.

Weighting data. A weight variable may be specified for each plot. Both V- and R-variables with decimal places are multiplied by a scale factor in order to obtain integer values. See "Input Dataset" section below.

When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. The univariate statistics which appear at the beginning of the results, immediately following the dictionary, are based on all cases which have valid data on each variable considered singly. For the plots themselves, the program eliminates cases which have missing

data on either or both of the variables in a particular plot. This pair-wise deletion also affects univariate and bivariate statistics which are printed at the top of each plot.

35.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Univariate statistics. The following are printed for each variable referenced, including plot filter and weight variables: minimum and maximum values, mean and standard deviation, and the number of cases with valid data values.

Key to plot coding scheme. A table showing the correspondence between the actual frequencies and the codes used in the plots.

Plot and statistics. For each plot requested, a 8 1/2 inch by 12 inch scatter diagram is printed. Univariate statistics (means, standard deviations) and bivariate statistics (Pearson's r , the regression constant A, and the regression unstandardized coefficient B) are printed at the top of the plot.

35.4 Input Dataset

The input is a Data file described by an IDAMS dictionary. All analysis and plot filter variables must be numeric; integer or decimal valued. Variables with decimals are multiplied by a scale factor in order to obtain integer values. This factor is calculated as 10^n where n is the number of decimals taken from the dictionary for V-variables and from the NDEC parameter for R-variables; it is printed for each variable.

35.5 Setup Structure

```
$RUN SCAT

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Plot specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

35.6 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-4 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V21=6 AND V37=5
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  STUDY 600. JULY 16, 1999. AGE BY HEIGHT FOR SUBSAMPLE 3
   ```

3. **Parameters** (mandatory). For selecting program options. New parameters are preceded by an asterisk.

   ```
   Example:  BADD=MD2
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   BADDATA=**STOP**/SKIP/MD1/MD2
   Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

   MAXCASES=n
   The maximum number of cases (after filtering) to be used from the input file.
   Default: All cases will be used.

   MDVALUES=**BOTH**/MD1/MD2/NONE
   Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

   * NDEC=**0**/n
   Number of decimals (maximum 4) to be retained for R-variables.

   PRINT=CDICT/DICT

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.

4. **Plot specifications.** One set for each plot. The coding rules are the same as for parameters. Each plot specification must begin on a new line.

   ```
   Example:  X=V3   Y=R17   FILTER=(V3,1,1)
   ```

   X=variable number
   Variable number of the X variable.

   Y=variable number
   Variable number of the Y variable.

   WEIGHT=variable number
   The weight variable number if the data are to be weighted.

FILTER=(variable number, minimum valid code, maximum valid code)
: Plot filter. Only those cases where the value of the filter variable is greater than or equal to the minimum code, and less than or equal to the maximum code, will be entered into the plot. For example, to specify that only cases with codes 0-40 on variable 6 are to be included, specify: FILTER=(V6,0,40).

HORIZAXIS=**MAXRANGE**/X
: MAXR Plot the variable with the greatest range along the horizontal axis.
X Plot always the X variable along the horizontal axis.

35.7 Restrictions

1. Not more than 50 variables can be used in one execution of the program. This maximum includes everything: X and Y variables, plot filter variables, weight and variables used in Recode statements.
2. No limit to the number of plots but SCAT produces only 5 plots for each pass of the input data.

35.8 Example

Generation of two plots (weighted by variable V100 and unweighted) repeated for three different subsets of data.

```
$RUN SCAT
$FILES
PRINT  = SCAT1.LST
DICTIN = MY.DIC                          input dictionary file
DATAIN = MY.DAT                          input data file
$SETUP
GENERATION OF TWO PLOTS REPEATED FOR EACH SUBSET OF DATA
*                                (default values taken for all parameters)
X=V21  Y=V3  FILTER=(V5,1,2)
X=V21  Y=V3  FILTER=(V5,1,2)  WEIGHT=V100
X=V21  Y=V3  FILTER=(V5,3,3)
X=V21  Y=V3  FILTER=(V5,3,3)  WEIGHT=V100
X=V21  Y=V3  FILTER=(V5,4,7)
X=V21  Y=V3  FILTER=(V5,4,7)  WEIGHT=V100
```

Chapter 36

Searching for Structure (SEARCH)

36.1 General Description

SEARCH is a binary segmentation procedure used to develop a predictive model for dependent variable(s). It searches among a set of predictor variables for those predictors which most increase the researcher's ability to account for the variance or for the distribution of a dependent variable. The question "what dichotomous split on which single predictor variable will give us a maximum improvement in our ability to predict values of the dependent variable?", embedded in an iterative scheme, is the basis for the algorithm used in this program.

SEARCH divides the sample, through a series of binary splits, into mutually exclusive series of subgroups. The subgroups are chosen so that, at each step in the procedure, the split into the two new subgroups accounts for more of the variance or the distribution (reduces the predictive error more) than a split into any other pair of subgroups.

SEARCH can perform the following functions:

* Maximize differences in group means, group regression lines, or distributions (maximum likelihood chi-square criterion).
* Rank the predictors to give them preference in the partitioning.
* Sacrifice explanatory power for symmetry.
* Start after a specified partial tree structure has been generated.

Generating a residuals dataset. Residuals may be computed and output as a data file described by an IDAMS dictionary. See the "Output Residuals Dataset" section for details on the content.

36.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. The dependent variable(s) are specified in the parameter DEPVAR, and the predictors are specified in the parameter VARS on predictor statements.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. Cases with missing data in a continuous dependent variable or a covariate are deleted automatically. Cases with missing data in a categorical dependent variable can be excluded by using a filter statement or by specifying valid codes with the DEPVAR parameter. Cases with missing data in the predictor variables are not automatically excluded. However, the filter statement and/or the CODES parameter may be used for this purpose.

36.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Outliers. (Optional: see the parameter PRINT). Outliers with the ID variable values and the dependent variable values.

Trace. (Optional: see the parameter PRINT, TRACE and FULLTRACE options). The trace of splits for each predictor for each split containing: the candidate groups for splitting, the group selected for splitting, all eligible splits for each predictor, the best split for each predictor and the split-on group.

Analysis summary containing the analysis of variance or distribution, the split summary and the summary of final groups.

Predictor summary tables. (Optional: see the parameter PRINT, TABLE, FIRST and FINAL options). The first group tables (PRINT=FIRST), the final group tables (PRINT=FINAL) or all groups' tables (PRINT=TABLE) containing summary of best splits for each predictor for each group. The tables are printed in reverse group order, i.e. last group first.

Tree diagram. (Optional: see the parameter PRINT). Hierarchical tree diagram. Each node (box) of the tree contains: group number, number of cases (N), split number, predictor variable number, mean of dependent variable (for means analysis), and mean of dependent variable and covariate, and slope (for regression analysis).

36.4 Output Residuals Dataset

Residuals can optionally be output in the form of a data file described by an IDAMS dictionary. (See the parameter WRITE). For means and regression analysis, and chi-square analysis with multiple dependent variables, each output record contains: an ID variable, the group variable, dependent variable(s), predicted (calculated) dependent variable(s), residual(s), and a weight, if any.

For chi-square analysis with one categorical dependent variable, it contains: an ID variable, the group variable, the first category of the dependent variable, the predicted (calculated) first category of the dependent variable, the residual for the first category of the dependent variable, the second category of the dependent variable, the predicted (calculated) second category of the dependent variable, the residual for the second category of the dependent variable, etc., and a weight, if any.

The characteristics of the output variables are as follows:

	Variable No.	Name	Field Width	No. of Decimals	MD1 Code
(ID variable)	1	same as input	*	0	same as input
(group variable)	2	Group variable	3	0	999
(dependent var 1)	3	same as input	*	**	same as input
(predicted var 1)	4	same as input cal	7	***	9999999
(residual for var 1)	5	same as input res	7	***	9999999
(dependent var 2)	6	same as input	*	**	same as input
(predicted var 2)	7	same as input cal	7	***	9999999
(residual for var 2)	8	same as input res	7	***	9999999
...	.	...	.	...	...
(weight-if weighted)	n	same as input	*	**	same as input

* transferred from input dictionary for V variables or 7 for R variables

** transferred from input dictionary for V variables or 2 for R variables

*** 6 plus no. of decimals for dependent variable minus width of dependent variable; if this is negative, then 0.

If the calculated value or residual exceeds the allocated field width, it is replaced by MD1 code.

36.5 Input Dataset

The input is a data file described by an IDAMS dictionary. All variables used for analysis must be numeric; they can be integer or decimal valued. The dependent variable may be continuous or categorical. Predictor variables may be ordinal or categorical. The case ID variable can be alphabetic.

36.6 Setup Structure

```
$RUN SEARCH

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Predictor specifications
     5. Predefined split specifications (optional)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output residuals dictionary
DATAyyyy   output residuals data
PRINT      results (default  IDAMS.LST)
```

36.7 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-5 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V3=5
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  SEARCHING FOR STRUCTURE
   ```

3. **Parameters** (mandatory). For selecting program options.

```
Example:  DEPV=V5
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

ANALYSIS=**MEAN**/REGRESSION/CHI

MEAN	Means analysis.
REGR	Regression analysis.
CHI	Chi-square analysis. With a single dependent variable, the default list of codes 0-9 will be used and no missing data verification will be made.

DEPVAR=variable number/(variable list)
The dependent variable or variables. Note that a list of variables can be provided only when ANALYSIS=CHI is specified.
No default.

CODES=(list of codes)
A list of codes may only be supplied for ANALYSIS=CHI and one dependent variable. Note that in this case no missing data verification is made for the dependent variable and only cases with the codes listed are used in analysis.

COVAR=variable number
The covariate variable number. Must be supplied for ANALYSIS=REGR.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

MINCASES=**25**/n
Minimum number of cases in one group.

MAXPARTITIONS=**25**/n
Maximum number of partitions.

SYMMETRY=**0**/n
The amount of explanatory power one is willing to lose in order to have symmetry, expressed as a percentage.

EXPL=**0.8**/n
Minimum increase in explanatory power required for a split, expressed as a percentage.

OUTDISTANCE=**5**/n
Number of standard deviations from the parent-group mean defining an outlier. Note that outliers are reported if PRINT=OUTL is specified, but they are not excluded from analysis.

IDVAR=variable number
Variable to be output with residuals and/or printed with each case classified as an outlier.

WRITE=RESIDUALS/CALCULATED/BOTH
Residuals and/or calculated values are to be written out as an IDAMS dataset.

RESI	Output residual values only.
CALC	Output calculated values only.
BOTH	Output both calculated values and residuals.

OUTFILE=**OUT**/yyyy
Applicable only if WRITE specified.
A 1-4 character ddname suffix for the residuals output dictionary and data files.
Default ddnames: DICTOUT, DATAOUT.

PRINT=(CDICT/DICT, TRACE, FULLTRACE, TABLE, FIRST, FINAL, TREE, OUTLIERS)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
TRAC	Print the trace of splits for each predictor for each split.
FULL	Print the full trace of splits for each predictor, including eligible but suboptimal splits.
TABL	Print the predictor summary tables for all the groups.
FIRS	Print the predictor summary tables for the first group.
FINA	Print the predictor summary tables for the final groups.
TREE	Print the hierarchical tree diagram.
OUTL	Print the outliers with ID variable and dependent variable values.

4. **Predictor specifications** (mandatory). Supply one set of parameters for each group of predictors which may be described with the same parameter values. The coding rules are the same as for parameters. Each predictor specification must begin on a new line.

```
Example:  VARS=(V8,V9) TYPE=F
```

VARS=(variable list)
Predictor variables to which the other parameters apply.
No default.

TYPE=**M**/F/S
The predictor constraint.

M	Predictors are considered to be "monotonic", i.e. the codes of the predictors are to be kept adjacent during the partition scan.
F	Predictor codes are considered to be "free".
S	Predictor codes will be "selected" and separated from the remaining codes in forming trial partitions.

CODES=**(0-9)**/maxcode/(list of codes)
Either the value of the largest acceptable code or a list of acceptable codes. The codes may range from 0 to 31. Cases with codes outside the range 0 to 31 are always discarded.

RANK=n
Assigned rank. If ranking is desired, assign a predictor rank of 0 to 9. A zero rank indicates that statistics are to be computed for the predictors, but they are not to be used in the partitioning.

5. **Predefined split specifications** (optional). If predefined splits are desired, supply one set of parameters for each predefined split. The coding rules are the same as for parameters. Each predefined split specification must begin on a new line.

```
Example:  GNUM=1  VAR=V18  CODES=(1-3)
```

GNUM=n
Number of the group to be split. Groups are specified in ascending order, where the entire original sample is group 1. Each set of parameters forms two new groups.
No default.

VAR=variable number
Predictor variable used to make the split.
No default.

CODES=(list of codes)
List of the predictor codes defining the first subgroup. All other codes will belong to the second subgroup.
No default.

36.8 Restrictions

1. Minimum number of cases required is 2 * MINCASES.
2. Maximum number of predictors is 100.
3. Maximum predictor value is 31.
4. Maximum number of categorical variable codes is 400.
5. Maximum number of predefined splits is 49.
6. If the ID variable is alphabetic with width > 4, only the first four characters are used.

36.9 Examples

Example 1. Means analysis with five predictor variables; minimum of 10 cases per group are requested; outliers of more than 3 standard deviations from the parent group mean are reported; cases are identified by the variable V1.

```
$RUN SEARCH
$FILES
PRINT   = SEARCH1.LST
DICTIN  = STUDY.DIC                    input dictionary file
DATAIN  = STUDY.DAT                    input data file
$SETUP
MEANS ANALYSIS  -  FIVE PREDICTOR VARIABLES
DEPV=V4  MINC=10  OUTD=3  IDVAR=V1  PRINT=(TRACE,TREE,OUTL)
VARS=(V3-V5,V12)
VARS=V21  TYPE=F  CODES=(1-4)
```

Example 2. Regression analysis with six predictor variables; residuals and calculated values are to be computed and written into a dataset (cases are identified by variable V2).

```
$RUN SEARCH
$FILES
PRINT   = SEARCH2.LST
DICTIN  = STUDY.DIC                    input dictionary file
DATAIN  = STUDY.DAT                    input data file
DICTOUT = RESID.DIC                    dictionary file for residuals
DATAOUT = RESID.DAT                    data file for residuals
$SETUP
REGRESSION ANALYSIS  -  SIX PREDICTOR VARIABLES
ANAL=REGR  DEPV=V12  COVAR=V7  MINC=10  IDVAR=V2  -
     WRITE=BOTH  PRINT=(TRACE,TABLE,TREE)
VARS=(V3-V5,V18)
VARS=V22  TYPE=F
```

Example 3. Chi analysis with one dependent categorical variable and selected codes; the first two splits are predefined.

```
$RUN SEARCH
$FILES
DICTIN = STUDY.DIC                     input dictionary file
DATAIN = STUDY.DAT                     input data file
$SETUP
CHI ANALYSIS - ONE DEPENDENT CATEGORICAL VARIABLE, PREDEFINED SPLITS
ANAL=CHI  DEPV=V101  CODES=(1-5)  MINC=5  PRINT=(FINAL,TREE)
VARS=(V3,V8)  TYPE=S
GNUM=1  VAR=V8  CODES=3
GNUM=2  VAR=V3  CODES=(1,2)
```

Chapter 37

Univariate and Bivariate Tables (TABLES)

37.1 General Description

The main use of TABLES is to obtain univariate or bivariate frequency tables with optional row, column and corner percentages and optional univariate and bivariate statistics. Tables of mean values of a variable can also be obtained.

Both univariate/bivariate tables and bivariate statistics can be output to a file so that can be used with a report generating program, or can be input to GraphID or other packages such as EXCEL for graphical display.

Univariate tables. Both univariate frequencies and cumulative univariate frequencies may be generated for any number of input variables and may also be expressed as percentages of the weighted or unweighted total frequency. In addition, the mean of a cell variable can be obtained.

Bivariate tables. Any number of bivariate tables may be generated. In addition to the weighted and/or unweighted frequencies, a table may contain frequencies expressed as percentages based on the row marginals, column marginals or table total, and the mean of a cell variable. These various items may be placed in a single table with a possible six items per cell, or each may be obtained as a distinct table.

Univariate statistics. For univariate analyses, the following statistics are available: mean, mode, median, variance (unbiased), standard deviation, coefficient of variation, skewness and kurtosis. A quantile option (NTILE) is also available. Division into as few as three parts or as many as ten parts may be requested.

Bivariate statistics. For bivariate analyses, the following statistics can be requested:

- t-tests of means (assumes independent populations) between pairs of rows,
- chi-square, contingency coefficient and Cramer's V,
- Kendall's Taus, Gamma, Lambdas,
- S (numerator of the tau statistics and of gamma), its standard and normal deviations, and its variance,
- Spearman rho,
- Evidence Based Medicine (EBM) statistics,
- non-parametric tests: Wilcoxon, Mann-Whitney and Fisher.

Matrices of statistics. Matrices of any of the above bivariate statistics except tests, EBM statistics or statistics of S can be printed or written to a file. Corresponding matrices of weighted and/or unweighted n's can be produced.

3- and 4-way tables. These can be constructed by making use of the repetition and subsetting features. The repetition variable can be thought of as a control or panel variable. The subsetting feature can be used to further select cases for a particular group of tables.

Tables of sums. Tables in which the cells contain the sum of a dependent variable can be produced by specifying the dependent variable as the weight. E.g. specify WEIGHT=V208, where V208 represents a

respondents income, in order to get the total income of all respondents falling into a cell.

Note. The following options are available to control the appearance of the results:

A title may be specified for each set of tables.

Percentages and mean values, if requested, may be printed in separate tables.

The grid can be suppressed.

Rows which have no entries in a particular section of a large frequency table can be printed; tables with more than ten columns are printed in sections and the use of this "zero rows" option ensures that the various sections have the same number of rows (which is important if they are to be cut and pasted together).

37.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. In addition, local filters and repetition factors (called subset specifications) may be used to select a subset of cases for a particular table. For tables which are individually specified, the variable(s) to be used for the table are selected with the table specification parameters R and C. For sets of tables, variables are selected with the table specification parameters ROWVARS and COLVARS.

Transforming data. Recode statements may be used. Note that for R-variables, the number of decimals to be retained is specyfied by the NDEC parameter.

Weighting data. A weight variable may optionally be specified for each set of tables. Both V- and R-variables with decimal places are multiplied by a scale factor in order to obtain integer values. See "Input Dataset" section below. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data.

1. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data.
2. Univariate and bivariate frequencies are always printed for all codes in the data whether or not they represent missing data. To remove missing data from tables completely, a filter or a subset can be specified. Alternatively appropriate minimum and/or maximum values of row and column variable can be defined.
3. Cases with missing data may optionally be included in the computation of percentages and bivariate statistics. This can be done using the MDHANDLING table parameter.
4. Cases with missing data on a cell variable are always excluded from univariate and bivariate tables.
5. Cases with missing data are always excluded from the computation of univariate statistics.

37.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

A table of contents for the results. The contents shows each table produced and gives the page number where it is located. The following information is provided:

- row and column variable numbers (0 if none)
- variable number for the mean value - cell variable (0 if none)
- weight variable number (0 if none)
- row minimum and maximum values (0 if none)
- column minimum and maximum values (0 if none)

- filter name and repetition factor name
- percentages: row, column and total (T=requested, F=not requested)
- RMD: row-variable missing data (T=delete, F=do not delete)
- CMD: column-variable missing data (T=delete, F=do not delete)
- CHI: chi-square (T=requested, F=not requested)
- TAU: tau a, b or c (T=requested, F=not requested)
- GAM: gamma (T=requested, F=not requested)
- TEE: t-tests (T=requested, F=not requested)
- EXA: Fisher non-parametric test (T=requested, F=not requested)
- WIL: Wilcoxon non-parametric test (T=requested, F=not requested)
- MW: Mann-Whitney non-parametric test (T=requested, F=not requested)
- SPM: Spearman rho (T=requested, F=not requested)
- EBM: Evidence Based Medicine statistics (T=requested, F=not requested).

Tables which were requested using the PRINT=MATRIX or WRITE=MATRIX table parameters are not listed in the contents and are always printed first with negative page and table numbers.

Other tables are printed in the order of the table specifications except for tables for which only univariate statistics are requested; these are always grouped together and printed last.

Bivariate tables. Each bivariate table starts on a new page; a large table may take more than one page. Tables are printed with up to 10 columns and up to 16 rows per page depending on the number of items in each cell. Columns and rows are printed only for codes which actually appear in the data. Row and column totals, and cumulative marginal frequencies and percentages if requested, are printed around the edges of the table.

A large table is printed in vertical strips. For example, a table with 40 row codes and 40 column codes would normally be printed on 12 pages as indicated in the following diagram, where the numbers in the cells show the order in which the pages are printed:

	1st 10	2nd 10	3rd 10	4th 10	codes
1st 16 codes	1	4	7	10	
2nd 16 codes	2	5	8	11	
last 8 codes	3	6	9	12	

Bivariate statistics. (Optional: see the table parameter STATS).

t-tests. (Optional: see the table parameter STATS). If t-tests were requested, they and the means and standard deviations of the column variable for each row are printed on a separate page.

Matrices of bivariate statistics. (Optional: see the table parameter PRINT). The lower-left corner of the matrix is printed. Eight columns and 25 rows are printed per page.

Matrix of N's. (Optional: see the table parameter PRINT). This is printed in the same format as the corresponding statistical matrix.

Univariate tables. (Optional: see the table parameter CELLS). Normally each univariate table is printed beginning on a new page. Frequencies, percents and mean values of a variable, if requested, for ten codes are printed across the page.

Univariate statistics. (Optional: see the table parameter USTATS).

Quantiles. (Optional: see the table parameter NTILE). N-1 points are printed; e.g. if quartiles are requested, the parameter NTILE is set to 4 and 3 breakpoints will be printed.

Page numbers. These are of the form: ttt.rr.ppp where

ttt = table number
rr = repetition number (00 if no repetition used)
ppp = page number within the table.

37.4 Output Univariate/Bivariate Tables

Univariate and/or bivariate tables with statistics requested in the table parameter CELLS may be output to a file by specifying WRITE=TABLES. The tables are in the format of IDAMS rectangular matrix (see "Data in IDAMS" chapter). One matrix is output for each statistic requested. If a repetition factor is used, one matrix is output for each repetition.

Columns 21-80 on the matrix-descriptor record contain additional description of the matrix as follows:

21-40	Row variable name (for bivariate tables).
41-60	Column variable name.
61-80	Description of the values in the matrix.

Variable identification records (#R and #C) contain code values and code labels for the row and the column variable respectively.

The statistics are written as 80 character records according to a 7F10.2 Fortran format. Columns 73-80 contain an ID as follows:

73-76	Identification of the statistic: FREQ, UNFR, ROWP, COLP, TOTP or MEAN.
77-80	Table number.

Note that the missing data codes are not included in the matrix.

37.5 Output Bivariate Statistics Matrices

Selected statistics may be output to a file. If, for example, gammas and tau b's were selected, a matrix of gammas and a separate matrix of tau b's would be generated. Output matrices of bivariate statistics are requested by specifying WRITE=MATRIX and either ROWVARS or ROWVARS and COLVARS table parameters. If a repetition factor is used, one matrix is output for each repetition. The matrices are in the format of IDAMS square or rectangular matrices (see "Data in IDAMS" chapter). The values in the matrix are written with Fortran format 6F11.5. Columns 73-80 contain an ID as follows:

73-76	Identification of the statistic: TAUA, TAUB, TAUC, GAMM, LSYM, LRD, LCD, CHI, CRMV or RHO.
77-80	Table number.

Note. If only ROWVARS is provided, dummy means and standard deviations records are written, 2 records per 60 variables. The second format (#F) record in the dictionary specifies a format of 60I1 for these dummy records. This is so that the matrix conforms to the format of an IDAMS square matrix.

37.6 Input Dataset

The input is a data file described by an IDAMS dictionary. With the exception of variables used in the main filter, all the other variables used must be numeric.

In distributions and weights, variables (both V and R) with decimal places are multiplied by a scale factor in order to obtain integer values. The scale factor is calculated as 10^n where n is the number of decimals taken from the dictionary for V-variables and from the NDEC parameter for R-variables; it is printed for each variable.

Univariate statistics without distributions are calculated using the number of decimals specified in the dictionary for V-variables and taken from NDEC parameter for R-variables.

Fields containing non-numeric characters (including fields of blanks) can be tabulated by setting the parameter BADDATA to MD1 or MD2. See "The IDAMS Setup File" chapter.

37.7 Setup Structure

```
$RUN TABLES

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters
     4. Subset specifications (optional)
     5. TABLES
     6. Table specifications (repeated as required)

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data


Files:
FT02       output tables/matrices
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
PRINT      results (default  IDAMS.LST)
```

37.8 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further descriptions of the program control statements, items 1-3 and 6 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example:  INCLUDE V3=6
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

   ```
   Example:  FREQUENCY TABLES
   ```

3. **Parameters** (mandatory). For selecting program options. New parameters are preceded by an asterisk.

   ```
   Example:  BADDATA=SKIP
   ```

   INFILE=**IN**/xxxx
   A 1-4 character ddname suffix for the input Dictionary and Data files.
   Default ddnames: DICTIN, DATAIN.

   BADDATA=**STOP**/SKIP/MD1/MD2
   Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
: The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

MDVALUES=**BOTH**/MD1/MD2/NONE
: Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

* NDEC=**0**/n
: Number of decimals (maximum 4) to be retained for R-variables.

PRINT=(CDICT/DICT, TIME)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
TIME	Print the time after each table.

4. **Subset specifications** (optional). These statements permit selection of a subset of cases for a table or set of tables.

```
Example:  CLASS INCLUDE V8=1,2,3,-7,9
```

There are two types of subset specifications: local filters and repetition factors. Each has a different function, but their formats are very similar. One specification may be used as a local filter for one or more tables and as a repetition factor for other tables.

Rules for coding

Prototype: name statement

name
: Subset name. 1-8 alphanumeric characters beginning with a letter. This name must match exactly the name used on subsequent analysis specifications. Embedded blanks are not allowed. It is recommended that all names be left-justified.

statement
: Subset definition which follows the syntax of the standard IDAMS filter statement.

For repetition factors, only one variable may be specified in the expression.

The way local filters and repetition factors work is described below.

Local filters. A subset specification is identified as a local filter for a table or set of tables by specifying the subset name with the FILTER parameter. The local filter operates in the same manner as the standard filter except that it applies only to the table specification(s) in which it is referenced.

```
Example:  EDUCATN          INCLUDE V4=0-4,9 AND V5=1
          (subset name)    (expression)
```

In the example above, if EDUCATN is designated as a local filter on the table specification, the table would be produced including only cases coded 0, 1, 2, 3, 4 or 9 for V4 and 1 for V5.

Repetition factors. A subset specification is identified as a repetition factor for a table or set of tables by specifying the subset name with the REPE parameter. Only one variable may be given on a subset specification to be used as a repetition factor. Repetition factors permit the generation of 3-way tables where the variable used in the repetition factor can be considered as the control or panel variable. Using a repetition factor and a filter, 4-way tables may be produced.

INCLUDE expressions cause tables to be produced including cases for each value or range of values of the control variable used in the expression. Commas separate the values or ranges. Thus if there are n commas in the expression, n+1 tables will be produced.

```
Example:  EDUCATN          INCLUDE V4=0-4,9
          (subset name)    (expression)
```

In the above example, if EDUCATN is designated as a repetition factor, two tables will result: one including cases coded 0-4 for variable 4, and another including cases coded 9 for variable 4.

EXCLUDE may be used to produce tables with all values except those specified.

```
Example:  EDUCATN          EXCLUDE V1=1,4
          (subset name)    (expression)
```

In the above example, if EDUCATN is designated as a repetition factor, two tables will result: one including all values except 1 and another including all values except 4.

5. **TABLES.** The word TABLES on this line signals that table specifications follow. It must be included (in order to separate subset specifications from table specifications) and must appear only once.

6. **Table specifications.** Table specifications are used to describe the characteristics of the tables to be produced. The coding rules are the same as for parameters. Each set of table specifications must start on a new line.

```
Examples:

R=(V6,1,8) CELLS=FREQS                (One univariate table).
R=(V6,1,8)  C=(V9,0,4) -              (One bivariate table with repetition
REPE=SEX  CELLS=(ROWP,FREQS)           factor, i.e. 3-way table).
ROWV=(V5-V9) CELLS=FREQS USTA=MEAN    (Set of univariate tables).
ROWV=(V3,V5) COLV=(V21-V31)  -        (Set of bivariate tables).
  R=(0,1,8) C=(0,1,99)
```

ROWVARS=(variable list)
: List of variables for which univariate tables are required or to be used as the rows in bivariate tables.

COLVARS=(variable list)
: List of variables to be used as columns for bivariate tables.

R=(var, rmin, rmax)

var	Row or univariate variable number for a single table. To supply minimum and maximum values for a set of tables, set the variable number to zero, e.g. R=(0,1,5); in this case the minimum and maximum codes apply to all variables in the ROWVARS parameter.
rmin	Minimum code of the row variable(s) for statistical and percent calculations.
rmax	Maximum code of the row variable(s) for statistical and percent calculations.

If either rmin or rmax is specified, both must be specified. If only the variable number is specified, minimum and maximum values are not applied.

C=(var, cmin, cmax)

var	Column variable number for a single bivariate table. To supply minimum and maximum values for a set of tables, set the variable number to zero, e.g. C=(0,2,5); in this case, the minimum and maximum codes apply to all variables in the COLVARS parameter.
cmin	Minimum code of the column variable(s) for statistical and percent calculations.
cmax	Maximum code of the column variable(s) for statistical and percent calculations.

If either cmin or cmax is specified, both must be specified. If only the variable number is specified, minimum or maximum values are not applied.

TITLE='table title'
Title to be printed at the top of each table in this set.
Default: No table title.

CELLS=(ROWPCT, COLPCT, TOTPCT, **FREQS**/NOFREQS, UNWFREQS, MEAN)
Contents of cells for tables when PRINT=TABLES or WRITE=TABLES specified.

ROWP	Percentages for univariate tables or percentages based on row totals for bivariate tables.
COLP	Percentages based on column totals in bivariate tables.
TOTP	Percentages based on grand total in bivariate tables.
FREQ	Weighted frequency counts (same as unweighted if WEIGHT not specified).
UNWF	Unweighted frequency counts.
MEAN	Mean of variable specified by VARCELL.

VARCELL=variable number
Variable number of the variable for which mean value is to be computed for each cell in the table.

MDHANDLING=**ALL**/R/C/NONE
Indicates which missing data values should be excluded from statistics and percent calculations.

ALL	Delete all missing data values.
R	Delete missing data values of row-variables.
C	Delete missing data values of column-variables.
NONE	Do not delete missing data. Note: missing data cases are always excluded from univariate statistics.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

FILTER=xxxxxxxx
The 1-8 character name of the subset specification to be used as a local filter. Enclose the name in primes if it contains any non-alphanumeric characters. If the name does not match with any subset specification, the table will be skipped. Upper case letters should be used in order to match the name on the subset specification which is automatically converted to upper case.

REPE=xxxxxxxx
The 1-8 character name of the subset specification to be used as a repetition factor. Enclose the name in primes if it contains any non-alphanumeric characters. If the name does not match with any subset specification, the table will be skipped. Tables will be repeated for each group of cases specified. Upper case letters should be used in order to match the name on the subset specification which is automatically converted to upper case.

USTATS=(MEANSD, MEDMOD)
(Univariate tables only).

MEAN	Print mean, minimum, maximum, variance (unbiased), standard deviation, coefficient of variation, skewness, kurtosis, weighted and unweighted total number of cases.
MEDM	Print median and mode (if there are ties, numerically smallest value is selected).

NTILE=n
(Univariate tables only).
The n is the number of quantiles to be calculated; it must be in the range 3-10.

STATS=(CHI, CV, CC, LRD, LCD, LSYM, SPMR, GAMMA, TAUA, TAUB, TAUC, EBMSTAT, WILC, MW, FISHER, T)
If any bivariate statistics are to be printed or output supply the STAT parameter with each of the statistics desired.

Bivariate tables and matrix output

CHI Chi-square. (If MATRIX is not requested, the selection of CHI, CV or CC will cause all three to be computed).
CV Cramer's V.
CC Contingency coefficient.
LRD Lambda, row variable is the dependent variable. (If MATRIX is not requested, the selection of any of the lambdas will cause all three to be computed).
LCD Lambda, column variable is the dependent variable.
LSYM Lambda, symmetric.
SPMR Spearman rho statistic.
GAMM Gamma statistic.
TAUA Tau a statistic. (If MATRIX is not requested, the selection of any of the three taus will cause all three to be computed).
TAUB Tau b statistic.
TAUC Tau c statistic.

Bivariate tables only

EBMS Evidence Based Medicine statistics.
WILC Wilcoxon signed ranks test.
MW Mann-Whitney test.
FISH Fisher exact test.
T t-tests between all combinations of rows, up to a limit of 50 rows.

DECPCT=**2**/n

Number of decimals, maximum 4, printed for percentages.

DECSTATS=**2**/n

Number of decimals printed for mean, median, taus, gamma, lambdas, and chi-square statistics. All other statistics will be printed with 2+n decimals (i.e. default of 4).

WRITE=MATRIX/TABLES

If an output file is to be generated, supply the WRITE parameter and the type of output.

MATR Output the matrices of selected statistics.
If the ROWVARS parameter is specified produce a square matrix for each statistic requested by the STATS parameter using all pairings of the variables appearing in the list.
If the ROWVARS and COLVARS parameters are specified produce a rectangular matrix for each statistic requested by the STATS parameter using each variable appearing in the ROWVARS list paired with each variable appearing in the COLVARS list.
TABL Output the tables of statistics requested with the CELLS parameter.

PRINT=(**TABLES**/NOTABLES, SEPARATE, ZEROS, CUM, **GRID**/NOGRID,
N, WTDN, MATRIX)

Options relevant to univariate/bivariate tables only.

TABL Print tables with items specified by CELLS.
SEPA Print each item specified in CELLS as a separate table.
ZERO Keep rows with zero marginals in results. (Applicable only if table has more than 10 columns and hence must be printed in strips).
CUM Print cumulative row and column marginal frequencies and percentages. If data are weighted, figures are computed on weighted frequencies only.
GRID Print grid around cells of bivariate tables.
NOGR Suppress grid around cells of bivariate tables.

Options relevant with WRITE=MATRIX only.

N Print matrix of n's for matrices of statistics requested.
WTDN Print matrix of weighted n's for matrices of statistics requested.
MATR Print matrices of statistics specified under STATS.

37.9 Restrictions

1. The maximum number of variables for univariate frequencies is 400.
2. The combination of variables and subset specifications is subject to the restriction:

   $5NV + 107NF < 8499$

   where NF is the number of subset specifications and NV is the number of variables.
3. Code values for univariate tables must be in the range -2,147,483,648 to 2,147,483,647.
4. Code values for bivariate tables must be in the range -32,768 to 32,767. Any code values outside this range are automatically recoded to the end points of the range, e.g. -40,000 will become -32,768 and 40,000 will become 32,767. Thus, on the bivariate table specification, 32,767 is the maximum "maximum value". (Note that a 5-digit variable with a missing data code of 99999 will have the missing data row labeled 32,767 on the results).
5. The maximum cumulative weighted or unweighted frequency for a table (and for any cell, row or column) is 2,147,483,647.
6. Table dimension maximums.

   Bivariate: 500 row codes, 500 column codes, 3000 cells with non-zero entities.
   Univariate: 3000 categories if frequencies, median/mode requested; otherwise, unlimited.
   Note: For a variable such as income, if there are more than 3000 unique income values, one cannot get a median or mode without first bracketing the variable.
7. Non-integer V-variable values in distributions and in weights are treated as if the decimal point were absent; a scale factor is printed for each variable.
8. t-tests of means between rows are performed only on the first 50 rows of a table.
9. For bivariate statistical matrix output, the maximum number of variables that may be requested for a row or column is 95.
10. If output files for tables and matrices are both requested, these are output to the same physical file.
11. There is no way of labelling rows and columns of tables when recoded variables are used.

37.10 Example

In the example below, the following tables are requested:

1. Frequency counts for variables V201-V220.
2. Univariate statistics with no frequency tables for variables V54-V62 and V64. Means will have 1 decimal and other statistics 3 decimals.
3. Weighted and unweighted frequency counts and percentages with cumulative frequencies and percentages for variables V25-V30 and a grouped version of variable V7. Missing data cases are not to be excluded from the percentages or statistics. Median and mode statistics requested.
4. For the categories of the single variable V201, frequency counts and the mean of variable V54.
5. 8 bivariate tables (with row variables V25-V28 and column variables V29, V30) repeated by values 1 and 2 of variable V10 (sex), i.e. with sex as a panel (control) variable. Counts, row, column and total percentages will be in each cell. Chi-square and Taus statistics requested.
6. 3-way tables, using region (V3) grouped into 3 categories as the panel variable. Tables are restricted to male cases only (V10=1). Frequency counts and mean of variable V54 will appear in each cell.
7. A single weighted frequency count table, excluding cases where either the row variable and/or the column variable take the value 9.
8. Matrices of Tau A and Gamma statistics to be printed and written to a file for all pairs of variables V54-V62. A matrix of counts of valid cases for each pair of variables will also be printed.

```
    $RUN TABLES
    $FILES
    PRINT  = TABLES.LST
    FT02   = TREE.MAT                  matrices of statistics
    DICTIN = TREE.DIC                  input Dictionary file
    DATAIN = TREE.DAT                  input Data file
    $RECODE
        R7=BRAC(V7,0-15=1,16-25=2,26-35=3,36-45=4,46-98=5,99=9)
        NAME R7'GROUPED V7'
    $SETUP
    TABLE EXAMPLES
    BADDATA=MD1
    MALE     INCLUDE V10=1
    SEX      INCLUDE V10=1,2
    REGION   INCLUDE V3=1-2,3-4,5
    MD       EXCLUDE V19=9 OR V52=9
    TABLES
1.  ROWV=(V201-V220)  TITLE='Frequency counts'
2.  ROWV=(V54-V62,V64)  USTATS=MEANSD  PRINT=NOTABLES  DECSTAT=1
3.  ROWV=(V25-V30,R7)   USTATS=MEDMOD  CELLS=(FREQS,UNWFREQS,ROWP)  -
        WEIGHT=V9  PRINT=CUM  MDHAND=NONE
4.  R=(V201,1,3)  CELLS=(FREQS,MEAN)  VARCELL=V54
5.  ROWV=(V25-V28)  COLV=(V29-V30)  -
        CELLS=(FREQS,ROWP,COLP,TOTP)  STATS=(CHI,TAUA)  REPE=SEX
6.  ROWV=(V201-V203)  COLV=V206  -
        CELLS=(FREQS,MEAN)  VARCELL=V54  REPE=REGION  FILT=MALE
7.  R=V19  C=V52  WEIGHT=V9  FILT=MD
8.  ROWV=(V54-V62)  STATS=(TAUA,GAMMA)  PRINT=(MATRIX,N)  WRITE=MATRIX
```

Chapter 38

Typology and Ascending Classification (TYPOL)

38.1 General Description

TYPOL creates a classification variable summarizing a large number of variables. The use of an initial classification variable, defined "a priori" (key variable), or a random sample of cases, or a step-wise sample are allowed to constitute the initial core of groups. An iterative procedure refines the results by stabilizing the cores. The final groups constitute the categories of the classification variable looked for. The number of groups of the typology may be reduced using an algorithm of hierarchical ascending classification.

The **active variables** are the variables on the basis of which the grouping and regrouping of cases is performed. One can also look for the main statistics of other variables within the groups constructed according to the active variables. Such variables (having no influence on the construction of the groups) are called **passive variables**.

TYPOL accepts both quantitative and qualitative variables, the latter being treated as quantitative after full dichotomization of their respective categories, which results in the construction of as many dichotomized (1/0) variables as the number of categories of the qualitative variable. It is also possible to standardize the active variables (the quantitative variables, and the qualitative after dichotomization).

TYPOL operates in two steps:

1. **Building of an initial typology.** The program builds a typology of n groups, as requested by the user, from the cases characterized by a given number of variables (considered as being quantitative). The user may select the way an initial configuration is established (see INITIAL parameter), and also the type of distance (see DTYPE parameter) used by the program for calculating the distance between cases and groups.

2. **Further ascending classification** (optional). If the user wants a typology in fewer groups, the program- using an algorithm of hierarchical ascending classification- reduces one by one the number of groups up to the number specified by the user.

38.2 Standard IDAMS Features

Case and variable selection. The standard filter is available to select a subset of cases from the input data. The variables are specified with parameters.

Transforming data. Recode statements may be used.

Weighting data. A variable can be used to weight the input data; this weight variable may have integer or decimal values. When the value of the weight variable for a case is zero, negative, missing or non-numeric, then the case is always skipped; the number of cases so treated is printed.

Treatment of missing data. The MDVALUES parameter is available to indicate which missing data values, if any, are to be used to check for missing data. Cases with missing data in the quantitative variables can be excluded from the analysis (see MDHANDLING parameter).

38.3 Results

Input dictionary. (Optional: see the parameter PRINT). Variable descriptor records, and C-records if any, only for variables used in the execution.

Initial typology

Construction of an initial typology. (Optional: see the parameter PRINT).

The regrouping of initial groups, followed by a table of cross-reference numbers attributed to the groups before and after the constitution of the initial groups.

Table(s) showing the re-distribution of cases between one iteration and the following one, and giving the percentage of the total number of cases properly grouped.

Evolution of the percentage of explained variance from one iteration to the other.

Characteristics of distances by groups. The number of cases in each initial group of the typology, together with the mean value and the standard deviation of distances.

Classification of distances. (Optional: see the parameter PRINT). Table showing, within each group, the distribution of cases across fifteen continuous intervals, these intervals being:

different for each group (first table),

identical for all groups (second table).

Global characteristics of distances. The total number of cases, with the overall mean and standard deviation of distances.

Summary statistics. The mean, standard deviation and the variable weight for the quantitative variables and for categories of qualitative active variables.

Description of resulting typology. For each typology group, its number and the percentage of cases belonging to it are printed first. Then the statistics are provided, variable by variable, in the following order: (1) quantitative active variables; (2) quantitative passive variables; (3) qualitative active variables; (4) qualitative passive variables.

For each quantitative variable is given its amount of explained variance, its overall mean value and, within each group of the typology, its mean value and standard deviation.

For each category of the qualitative variable is given first its amount of variance explained and the percentage of cases belonging to it; then within each group of the typology are printed: vertically, the percentage of cases across the categories of the variable in the 1st line and horizontally, the percentage of cases across the groups of the typology (row percentages) in the 2nd line (optional: see the parameter PRINT).

Summary of the amount of variance explained by the typology. The following percentages of explained variance are given:

the variance explained by the most discriminant variables, i.e. those which taken altogether are responsible for eighty per cent of the explained variance,
the mean amount of variance explained by the active variables,
the mean amount of variance explained by all the variables together,
the mean amount of variance explained by the most discriminant variables together with the proportion of these variables.

Note: When qualitative variables appear in tables, the first 12 characters of the variable name are printed together with the code value identifying the category. When quantitative variables appear in tables, all 24 characters of the variable name are printed.

Ascending hierarchical classification

Table of square roots of displacements and distances calculated for each pair of groups. (Optional: see the parameter PRINT).

Table of regrouping No. 1. Summary statistics for the quantitative active variables and categories of qualitative active variables for groups involved in regroupment.

Description of new resulting typology. (Optional: see the parameter LEVELS). The same information as above.

Summary of the amount of variance explained by the new typology. The same information as above. Note here the mean amount of variance explained by the most discriminant variables before regrouping.

The summary of the ascending hierarchical classification is printed after each regroupment up to the number of groups specified by the user.

Three diagrams showing the percentage of explained variance as a function of the number of groups of the successive typologies, in turn for:
all the variables,
the active variables,
the variables explaining 80% of the variance before the regroupings took place.

Profiles of each group of the typology. (Optional: see the parameter PRINT). These profiles are printed and plotted for all the groups of the first resulting typology and then for the groups obtained at each regrouping.

Hierarchical tree is produced at the end.

38.4 Output Dataset

A "classification variable" dataset for the first resulting typology can be requested and is output in the form of a data file described by an IDAMS dictionary (see parameter WRITE and "Data in IDAMS" chapter). It contains the case ID variable, the transferred variables, the classification variable ("GROUP NUMBER") and, for each case, its distance multiplied by 1000 from each category of the classification variable, called "n GROUP DISTANCE". The variables are numbered starting from one and incrementing by one in the following order: case ID variable, transferred variables, classification variable and distance variables.

38.5 Output Configuration Matrix

An output configuration matrix may optionally be written in the form of an IDAMS rectangular matrix (see parameter WRITE). See "Data in IDAMS" chapter for a description of the format. This matrix provides, line by line, for each quantitative variable and for each category of qualitative active variables, its mean value across the groups and its overall standard deviation for the initial typology, i.e. before the regroupings take place. The elements of the matrix are written in 8F9.3 format. Dictionary records are written.

38.6 Input Dataset

The input is a Data file described by an IDAMS dictionary. All analysis variables must be numeric; they may be integer or decimal valued. The case ID variable and variables to be transferred can be alphabetic.

38.7 Input Configuration Matrix

The input configuration matrix must be in the form of an IDAMS rectangular matrix. See "Data in IDAMS" chapter for a description of the format. This matrix is optional and provides a starting configuration to be used in the computations. The statistics included should be mean values for the quantitative variables and proportions (not percentages) for the categories of qualitative variables (e.g. .180 instead of 18.0 per cent). A configuration matrix output by the program in a previous execution may serve as input configuration.

38.8 Setup Structure

```
$RUN TYPOL

$FILES
     File specifications

$RECODE (optional)
     Recode statements

$SETUP
     1. Filter (optional)
     2. Label
     3. Parameters

$DICT (conditional)
     Dictionary

$DATA (conditional)
     Data

$MATRIX (conditional)
     Input configuration matrix


Files:
FT02       output configuration matrix if WRITE=CONF specified
FT09       input configuration matrix if INIT=INCONF specified
           (omit if $MATRIX used)
DICTxxxx   input dictionary (omit if $DICT used)
DATAxxxx   input data (omit if $DATA used)
DICTyyyy   output dictionary if WRITE=DATA specified
DATAyyyy   output data if WRITE=DATA specified
PRINT      results (default  IDAMS.LST)
```

38.9 Program Control Statements

Refer to "The IDAMS Setup File" chapter for further description of the program control statements, items 1-3 below.

1. **Filter** (optional). Selects a subset of cases to be used in the execution.

   ```
   Example: INCLUDE V1=10-40,50
   ```

2. **Label** (mandatory). One line containing up to 80 characters to label the results.

```
Example: FIRST CONSTRUCTION OF CLASSIFICATION VARIABLE
```

3. **Parameters** (mandatory). For selecting program options.

```
Example: MDHAND=ALL AQNTV=(V12-V18) DTYP=EUCL  -
            PRINT=(GRAP,ROWP,DIST) INIG=5 FING=3
```

INFILE=**IN**/xxxx
A 1-4 character ddname suffix for the input Dictionary and Data files.
Default ddnames: DICTIN, DATAIN.

BADDATA=**STOP**/SKIP/MD1/MD2
Treatment of non-numeric data values. See "The IDAMS Setup File" chapter.

MAXCASES=n
The maximum number of cases (after filtering) to be used from the input file.
Default: All cases will be used.

AQNTVARS=(variable list)
A variable list specifying quantitative active variables.

PQNTVARS=(variable list)
A variable list specifying quantitative passive variables.

AQLTVARS=(variable list)
A variable list specifying qualitative active variables.

PQLTVARS=(variable list)
A variable list specifying qualitative passive variables.

MDVALUES=**BOTH**/MD1/MD2/NONE
Which missing data values are to be used for the variables accessed in this execution. See "The IDAMS Setup File" chapter.

MDHANDLING=**ALL**/QUALITATIVE/QUANTITATIVE

ALL	Cases with missing data values in quantitative variables will be skipped and missing data codes in qualitative variables will be excluded from analysis.
QUAL	Missing data values in qualitative variables will be excluded from analysis.
QUAN	Cases with missing data values in quantitative variables will be skipped.

REDUCE
Standardization of active variables, both quantitative and qualitative.

WEIGHT=variable number
The weight variable number if the data are to be weighted.

DTYPE=**CITY**/EUCLIDEAN/CHI

CITY	City block distance.
EUCL	Euclidean distance.
CHI	Chi-square distance.

Note: Concerning the choice of type of distance it is advisable to use:

- the City block distance when some active variables are qualitative and others are quantitative,

- the Euclidean distance when active variables are all quantitative (with standardization if they are not measured on the same scale),
- the Chi-square distance when active variables are all qualitative.

INIGROUP=n
: Number of initial groups. If a key variable is to serve as a basis for the typology, and if the number of initial groups specified here is greater than the maximum value of the key variable, the program corrects this automatically. Also, if there are certain categories with zero cases, the number of initial groups will be the number of non-empty categories.
No default.

FINGROUP=**1**/n
: Number of final groups.

INITIAL=**STEPWISE**/RANDOM/KEY/INCONF
: The way the initial configuration is established.
STEP Stepwise sample.
RAND Random sample.
KEY Profile of initial groups is created according to a key variable.
INCO An "a priori" profile of initial groups is given in an input configuration file.
Note: Variables included in the input configuration must correspond exactly to the variables provided with the AQNTV and/or AQLTV parameters.

STEP=**5**/n
: If stepwise sample of cases is requested (INIT=STEP), n is the length of the step.

NCASES=n
: If the random sample of cases is requested (INIT=RAND), n is the number of cases (unweighted) in the input file, or a good underestimation of it.
No default; must be specified if INIT=RAND.

KEY=variable number
: If a key variable is used to construct initial groups (INIT=KEY), this is the number of the key variable.
No default; must be specified if INIT=KEY.

ITERATIONS=**5**/n
: Maximum number of iterations for convergence of the group profile.

REGROUP=**DISPLACEMENT**/DISTANCE
: DISP Regrouping is based on minimum displacement.
DIST Regrouping is based on minimum distance.

WRITE=(DATA, CONFIG)
: DATA Create an IDAMS dataset containing the case ID variable, transferred variables, classification variable and distance variables.
CONF Output the configuration matrix into a file.

OUTFILE=**OUT**/yyyy
: A 1-4 character ddname suffix for the output Dictionary and Data files.
Default ddnames: DICTOUT, DATAOUT.

IDVAR=variable number
: Variable to be transferred to the output dataset to identify the cases.
Obligatory if WRITE=DATA specified.

TRANSVARS=(variable list)
Additional variables (up to 99) to be transferred to the output dataset.

LEVELS=(n1, n2, ...)
Print description of resulting typology for the number of groups specified.
Default: Description is printed after each regrouping.

PRINT=(CDICT/DICT, OUTCDICT/OUTDICT, INITIAL, TABLES, GRAPHIC, ROWPCT, DISTANCES)

CDIC	Print the input dictionary for the variables accessed with C-records if any.
DICT	Print the input dictionary without C-records.
OUTC	Print the output dictionary with C-records if any.
OUTD	Print the output dictionary without C-records.
INIT	Print history of initial typology construction.
TABL	Print two tables with classification of distances.
GRAP	Print the graphic of profiles.
ROWP	Print row percentages for categories of qualitative variables.
DIST	Print table of distances and displacements for each regrouping.

38.10 Restrictions

1. Maximum number of initial groups is 30.
2. Maximum total number of variables is 500, including weight variable, key variable, variables to be transferred, analysis variables (quantitative variables + number of categories for qualitative variables) and variables used temporarily in Recode statements.
3. If the ID variable or a variable to be transferred is alphabetic with width > 4, only the first four characters are used.
4. R-variables cannot be used as ID or as variables to be transferred.

38.11 Examples

Example 1. Creation of a classification variable summarizing 5 quantitative and 4 qualitative variables using the City block distance; initial configuration will be established by random selection of cases; classification starts with 6 groups and will terminate with 3 groups; regrouping will be based on minimum distance; missing data will be excluded from analysis.

```
$RUN TYPOL
$FILES
PRINT  = TYPOL1.LST
DICTIN = A.DIC                          input Dictionary file
DATAIN = A.DAT                          input Data file
$SETUP
SEARCHING FOR NUMBER OF CATEGORIES IN A CLASSIFICATION VARIABLE
AQNTV=(V114,V116,V118,V120,V122)  AQLTV=(V5-V7,V36)  REDU  -
   INIG=6  FING=3  INIT=RAND  NCAS=1200  -
   REGR=DIST  PRINT=(GRAP,ROWP,DIST)
```

Example 2. Generating a classification variable from Example 1 with 4 categories; the variable is to be written into a file; variables V18 and V34 are used as quantitative passive and variables V12 and V14 as qualitative passive.

```
$RUN TYPOL
$FILES
PRINT   = TYPOL2.LST
DICTIN  = A.DIC                          input Dictionary file
DATAIN  = A.DAT                          input Data file
DICTOUT = CLAS.DIC                       output Dictionary file
DATAOUT = CLAS.DAT                       output Data file
$SETUP
GENERATING A CLASSIFICATION VARIABLE
AQNTV=(V114,V116,V118,V120,V122)  AQLTV=(V5-V7,V36)  REDU  -
   PQNTV=(V18,V34)  PQLTV=(V12,V14)  -
   INIG=6  FING=4  INIT=RAND  NCAS=1200 -
   REGR=DIST  PRINT=(GRAP,ROWP) WRITE=DATA  IDVAR=V1
```

Part V

Interactive Data Analysis

Chapter 39

Multidimensional Tables and their Graphical Presentation

39.1 Overview

The interactive "Multidimensional Tables" component of WinIDAMS allows you to visualize and customize multidimensional tables with frequencies, row, column and total percentages, univariate statistics (sum, count, mean, maximum, minimum, variance, standard deviation) of additional variables, and bivariate statistics. Variables in rows and/or columns can either be nested (maximum 7 variables) or they can be put at the same level. Construction of a table can be repeated for each value of up to three "page" variables. Each page of the table can also be printed, or exported in free format (comma or tabulation character delimited) or in HTML format.

IDAMS datasets used as input must have the same name for the Dictionary and Data files with extensions .dic and .dat respectively.

Only one dataset can be used at a time, i.e. opening another dataset automatically closes the one being used.

39.2 Preparation of Analysis

Selection of data. A dataset selected for constructing multidimensional tables is available until it is changed when activating again the "Multidimensional Tables" component. The dialogue box lets you choose a Data file either from a list of recently used Data files (Recent) or from any folder (Existing). The Data folder of the current application is the default. Setting "Files of type:" to "IDAMS Data Files (*.dat)" displays only IDAMS Data files.

Selection of variables. Selection of a dataset for analysis calls the dialogue box for table definition. You are presented with a list of available variables and with four windows to specify variables for different purposes. Use Drag and Drop technique to move variables between and/or within required windows.

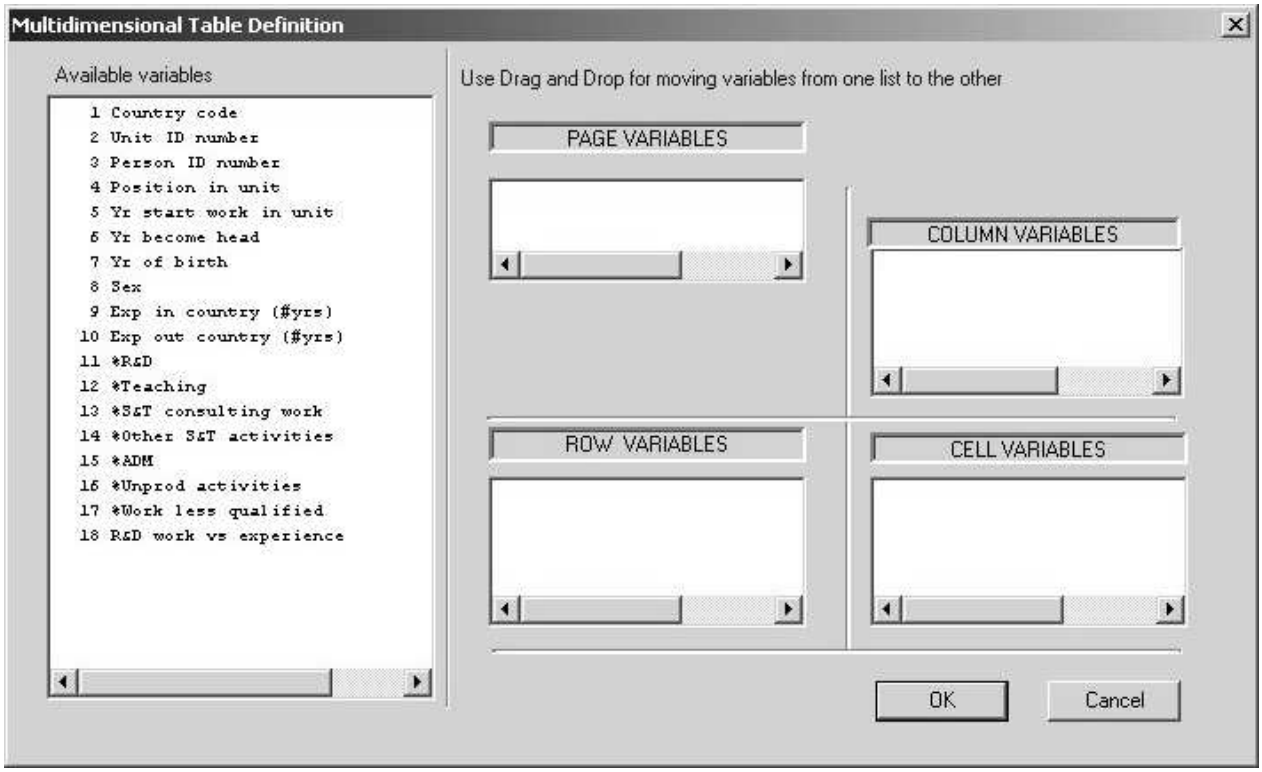
Page variables are used to construct separate pages of the table for each distinct value of each variable in turn, and for all cases taken together (Total page). Cases included on a particular page have all the same value on the page variable. Page variables are never nested. The order in which variables are specified determines the order in which pages are placed in the Table window.

Row variables are the variables whose values are used to define table rows. Their order determines the sequence of nesting use.

Column variables are the variables whose values are used to define table columns. Their order determines the sequence of nesting use.

Cell variables are variables whose values are used to calculate univariate statistics (e.g. mean) in the table

cells. The order in which they are specified determines the order of their appearance in the table. There may be up to 10 cell variables.



Nesting. If more than one row and/or column variable is specified, by default they are nested. To use them sequentially, at the same level, double-click on the variable in the row or column variable list and mark the option for treating at the same level. Note: This option is not available for the first variable in a list.

Percentages. Percentages in each cell (row, column or total) can be obtained by double-clicking on the last nested row variable in the table definition window and selecting the type of percentages required.

Univariate statistics. Different statistics (sum, count, mean, maximum, minimum, variance, standard deviation) for each of the cell variables can be obtained by double-clicking on the variable in the table definition window and marking the required statistic(s). Formulas for calculating mean, variance and standard deviation can be found in section "Univariate Statistics" of "Univariate and Bivariate Tables" chapter. However, they need to be adjusted since cases are not weighted.

Missing data treatment. The default missing data treatment is applied to the first construction of the table. Then, it can be changed using the menu Change.

Missing Data Values option is used to indicate which missing data values, if any, are to be used to check for missing data in row and column variables.

Both	Variable values will be checked against the MD1 codes and against the ranges of codes defined by MD2.
MD1	Variable values will be checked only against the MD1 codes.
MD2	Variable values will be checked only against the ranges of codes defined by MD2.
None	MD codes will not be used. All data values will be considered valid.

By default, both MD codes are used.

Missing Data Handling option is used to indicate which missing data values should be excluded from computation of percentages and bivariate statistics.

All	Delete all missing data values.
Row	Delete missing data values of row variables.
Column	Delete missing data values of column variables.
None	Do not delete missing data values.

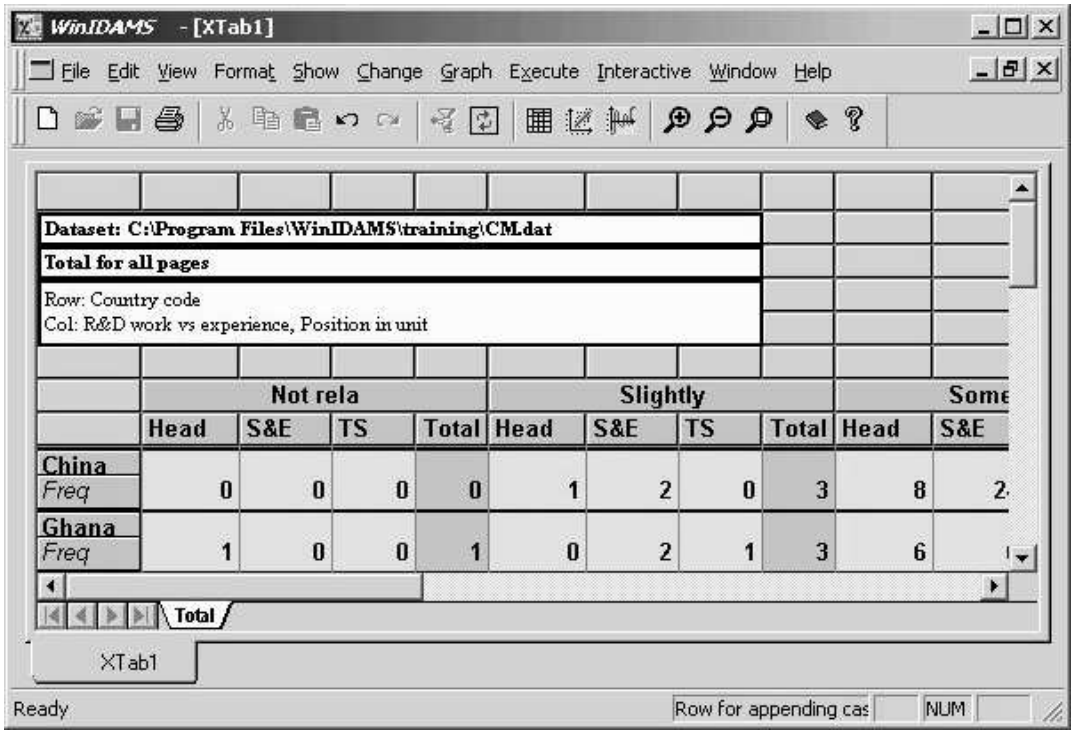
By default, all missing data values are deleted.

Note: Cases with missing data on cell variables are always excluded from calculation of univariate statistics. The exclusion is done cell by cell, separately for each cell variable. Thus, the number of valid cases may not be equal to the cell frequency. The statistic "Count" shows the number of valid cases.

Changing table definition. The menu command Change/Specification calls the dialogue box with the active table definition. You can change variables for analysis, their nesting as well as requests for percentages and univariate statistics. Clicking on OK replaces the active table by a new one.

39.3 Multidimensional Tables Window

After selection of variables and a click on OK, the Multidimensional Tables window appears in the WinIDAMS document window. By default, frequencies and mean values for all cell variables are displayed. If page variables are specified, code labels (or codes) of these variables are displayed on tabs at the bottom of the table. A particular page can be accessed by a click on the required label (code).


WinIDAMS - [XTab1]
File Edit View Format Show Change Graph Execute Interactive Window Help
Dataset: C:\Program Files\WinIDAMS\training\CM.dat
Total for all pages
Row: Country code
Col: R&D work vs experience, Position in unit
Not rela
Slightly
Some
Head S&E TS Total Head S&E TS Total Head S&E
China
Freq 0 0 0 0 1 2 0 3 8 2
Ghana
Freq 1 0 0 1 0 2 1 3 6
Total
XTab1
Ready
Row for appending cas
NUM

Changing the page appearance. The appearance of each page can be changed separately, the changes applying exclusively to the active page.

The following modifications are possible:

- *Increasing the font size* - use the menu command View/Zoom In or the toolbar button Zoom In.
- *Decreasing the font size* - use the menu command View/Zoom Out or the toolbar button Zoom Out.
- *Resetting default font size* - use the menu command View/100% or the toolbar button 100%.
- *Increasing/Decreasing the width of a column* - place the mouse cursor on the line which separates two columns in the column heading until it becomes a vertical bar with two arrows and move it to the right/left holding the left mouse button.
- *Minimizing the width of columns* - mark the required column(s) and use the menu command Format/Resize Columns.
- *Increasing/Decreasing the height of rows* - place the mouse cursor on the line which separates two rows in the row heading until it becomes a horizontal bar with two arrows and move it down/up holding the left mouse button.

- *Minimizing the height of rows* - mark the required row(s) and use the menu command Format/Resize Rows.
- *Hiding columns/rows* - decrease the width/height of a column/row to zero. To display back a hidden column/row, place the mouse cursor on the line where it is hidden in the column/row heading until it becomes a vertical/horizontal bar with two arrows and double-click the left mouse button.

In addition, the command Format/Style gives access to a number of **table formatting** possibilities such as: selection of fonts, size of fonts, colours, etc. for the active cell or for all cells in the active line.

Bivariate statistics. Bivariate statistics (Chi-square, Phi coefficient, contingency coefficient, Cramer's V, Taus, Gamma, Lambdas and Sormer's D) are computed for each table (each page). Use the menu command Show/Statistics to display them at the end of table. If needed, this operation should be repeated for each page separately. Formulas for calculating bivariate statistics can be found in section "Bivariate Statistics" of "Univariate and Bivariate Tables" chapter.

Note that statistics are calculated only when there is one row and one column variable.

Printing a table page. The whole contents of the active page or desired parts only can be printed using the File/Print command. If you want to print only some columns and/or rows, hide the other columns/rows first. The displayed columns/rows will be printed.

Exporting a table page. The whole contents of the active page or desired parts only can be exported in free format (comma or tabulation character delimited) or in HTML format. Use the File/Export command and select the required format. If you want to export only some columns and/or rows, hide the other columns/rows first. The displayed columns/rows will be exported.

39.4 Graphical Presentation of Univariate/Bivariate Tables

Frequencies displayed in a page of univariate/bivariate tables can be presented graphically using one of 24 graph styles at your disposal. Graph construction is initiated by the menu command Graph/Make. This command calls the dialogue box to select the graph style for the active page. In addition, you may ask to use logarithmic transformation of frequencies, and to provide a legend for colours and symbols used in the graph.

Projected graphics cannot be manipulated. However, they can be saved in one of the two formats, namely: JPEG file interchange format (.jpg) or Windows Bitmap format (.bmp), using the relevant commands in the File menu. They can also be copied to the Clipboard (the command Edit/Copy, toolbar button Copy or shortcut keys Ctrl/C) and passed to any text editor.

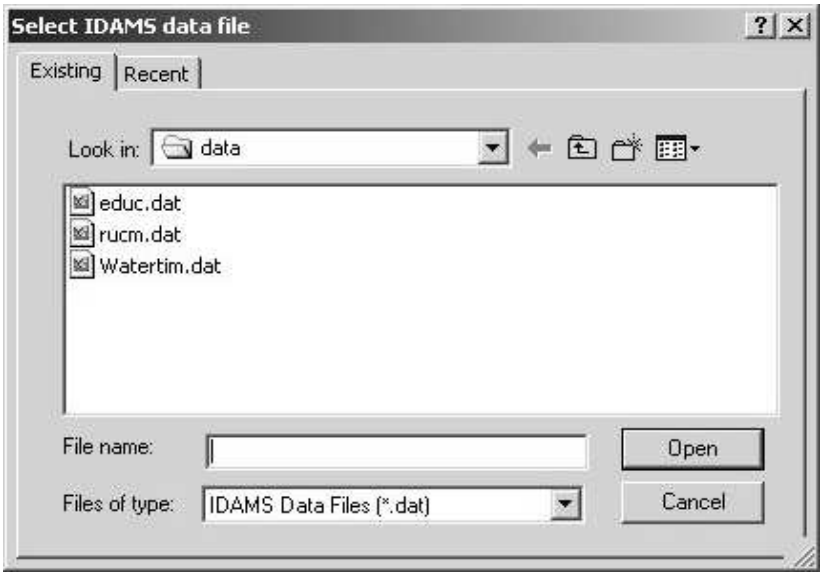
It should be noted here again that only frequencies from displayed rows and columns, i.e. not from rows and/or columns which have been hidden, are used for this presentation.

39.5 How to Make a Multidimensional Table

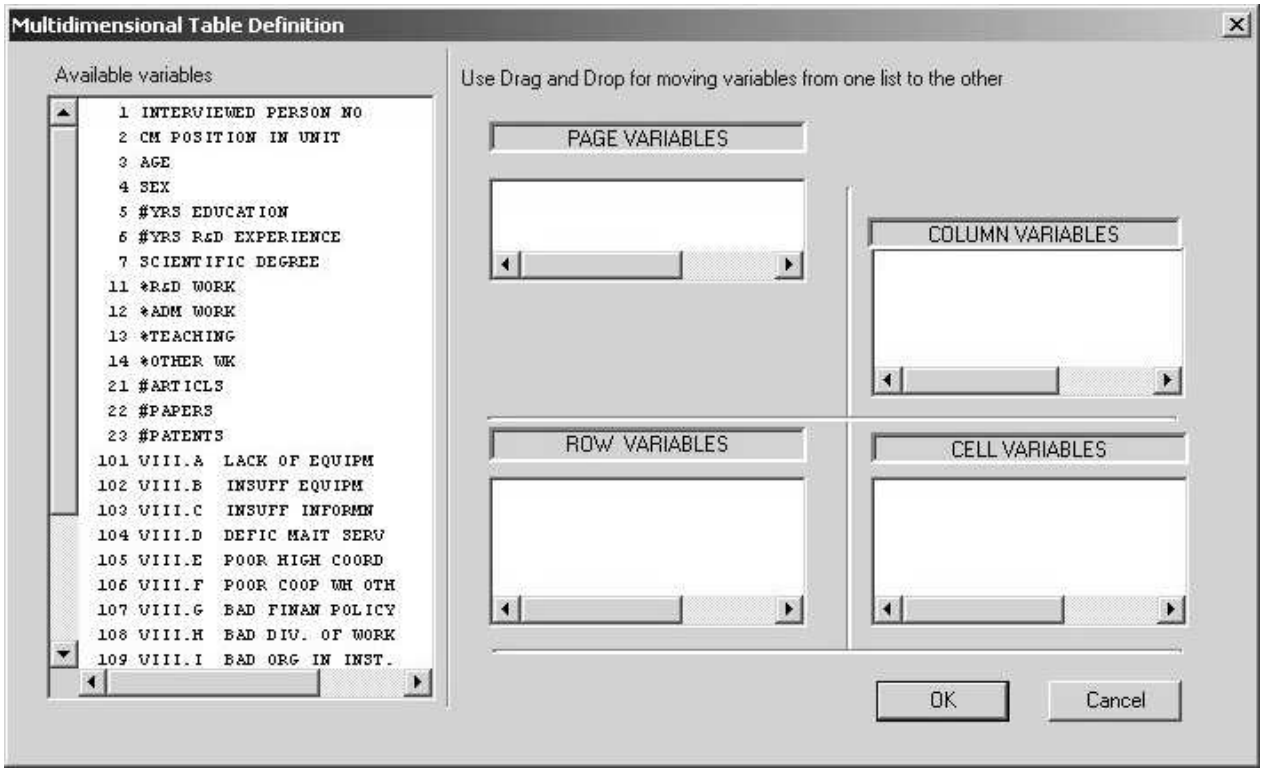
We will use the "rucm" dataset ("rucm.dic" is the Dictionary file and "rucm.dat" is the Data file) which is in the default Data folder and which is installed with WinIDAMS.

We will build a three-way table with two nested row variables ("SCIENTIFIC DEGREE" and "SEX"), one column variable ("CM POSITION IN UNIT") and one cell variable ("AGE") for which we will ask the mean, maximum and minimum.

- Click on Interactive/Multidimensional Tables. This command opens a dialogue for selecting an IDAMS Data file.



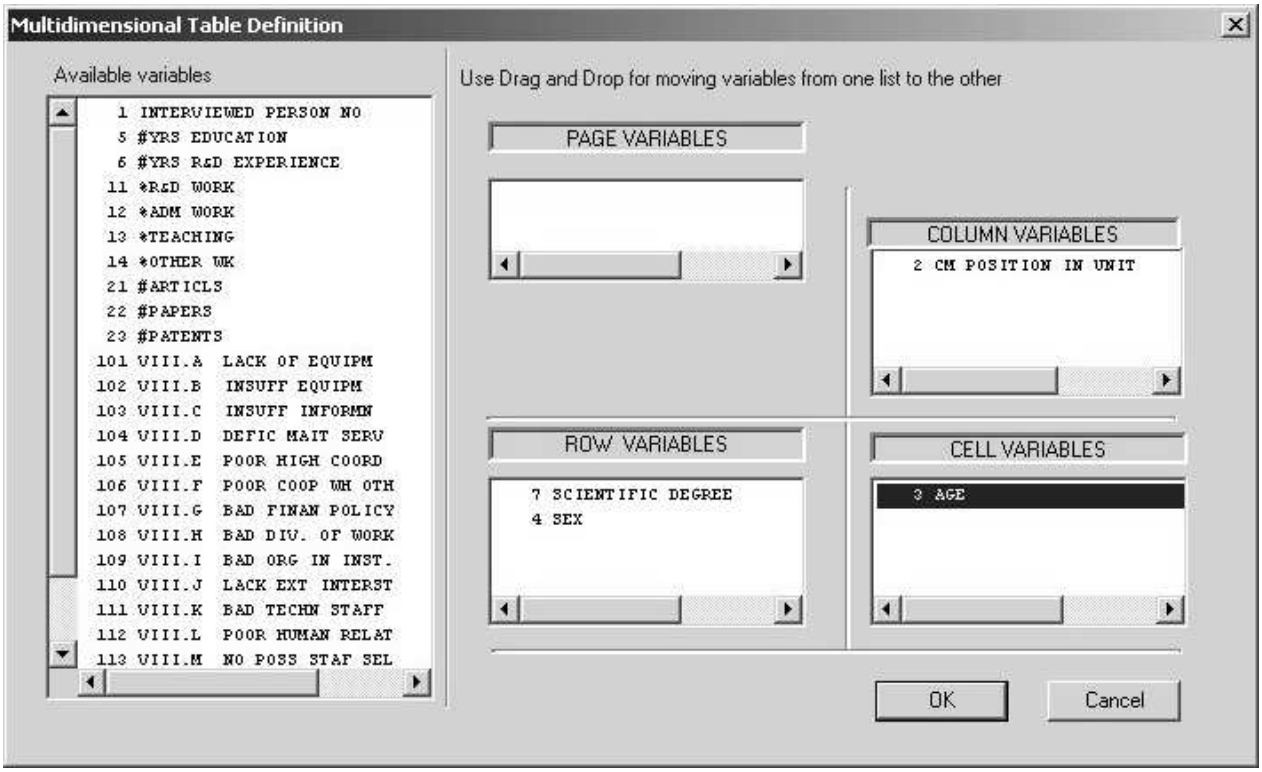
- Click on rucm.dic and Open. You now see a dialogue for specifying the variables that you want to use in the multidimensional table.



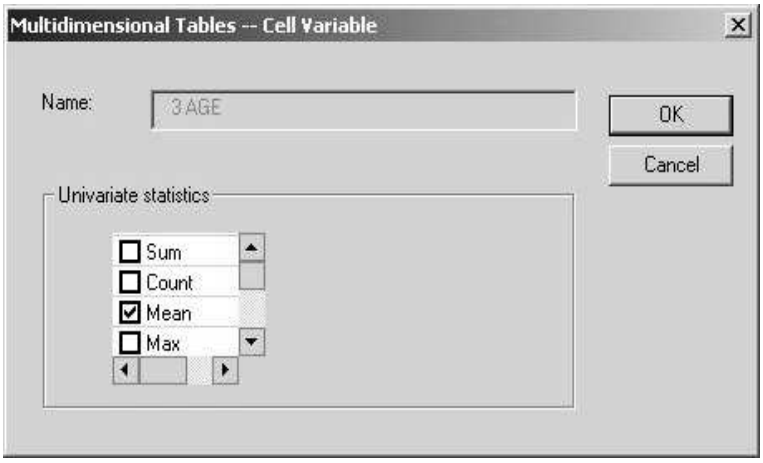
- Select variables “SCIENTIFIC DEGREE” and “SEX” as ROW VARIABLES, “CM POSITION IN UNIT” as COLUMN VARIABLE and “AGE” as CELL VARIABLE.

  Use the mouse Drag and Drop technique to move the variables (press the left mouse button on the variable you want to move, hold down the mouse button while you move the variable and release on the variable list where you want to move the variable). Several variables can be selected and moved simultaneously from one list to the other (hold down the Ctrl key when selecting).

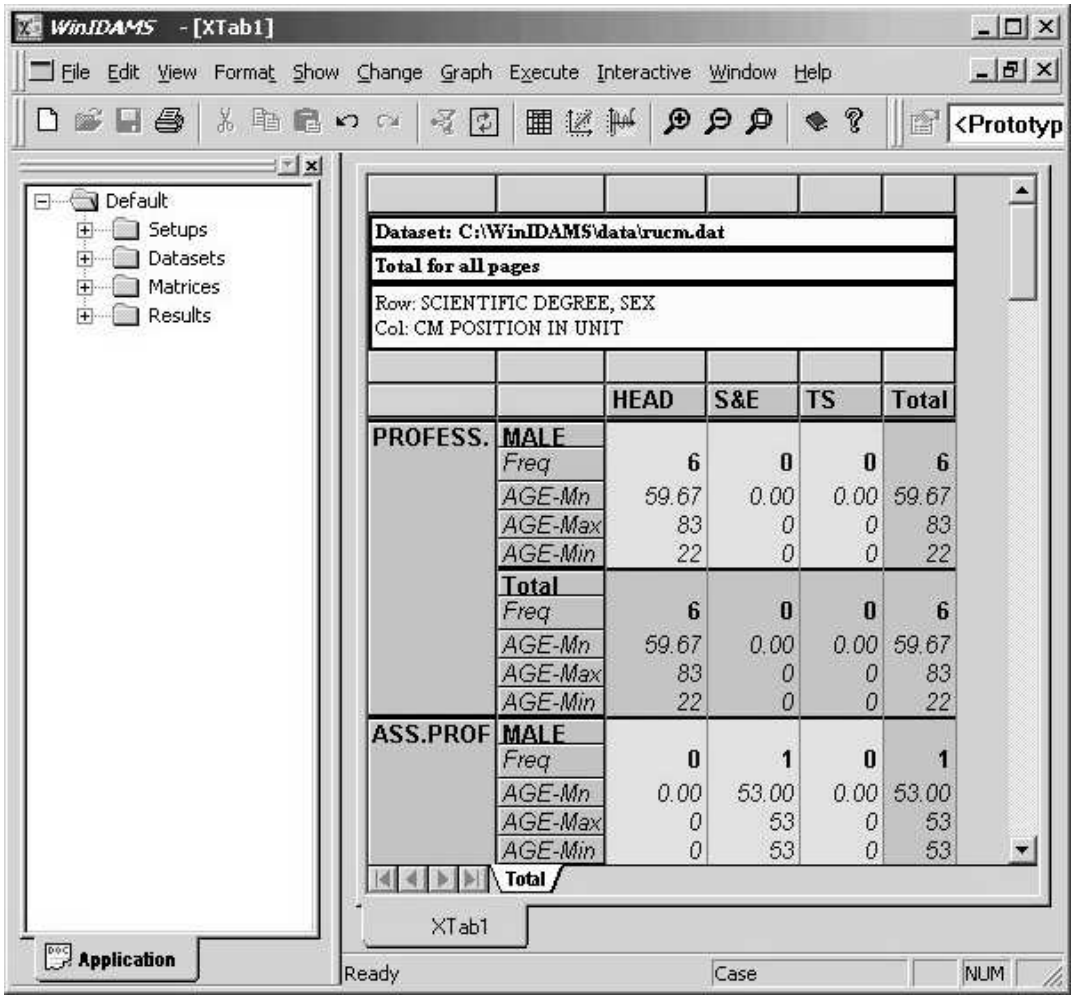
  The order of the variables in the ROW VARIABLES and COLUMN VARIABLES lists specifies, implicitly, the nesting order. The first variable in the list will be the outermost one. The variable order in a list can be modified using the Drag and Drop mouse technique inside the same list.



- After selecting the variables, the default options assigned to a variable can be changed by double-clicking on the variable. A double-click on the variable "AGE" in the CELL VARIABLES list opens the following dialogue:



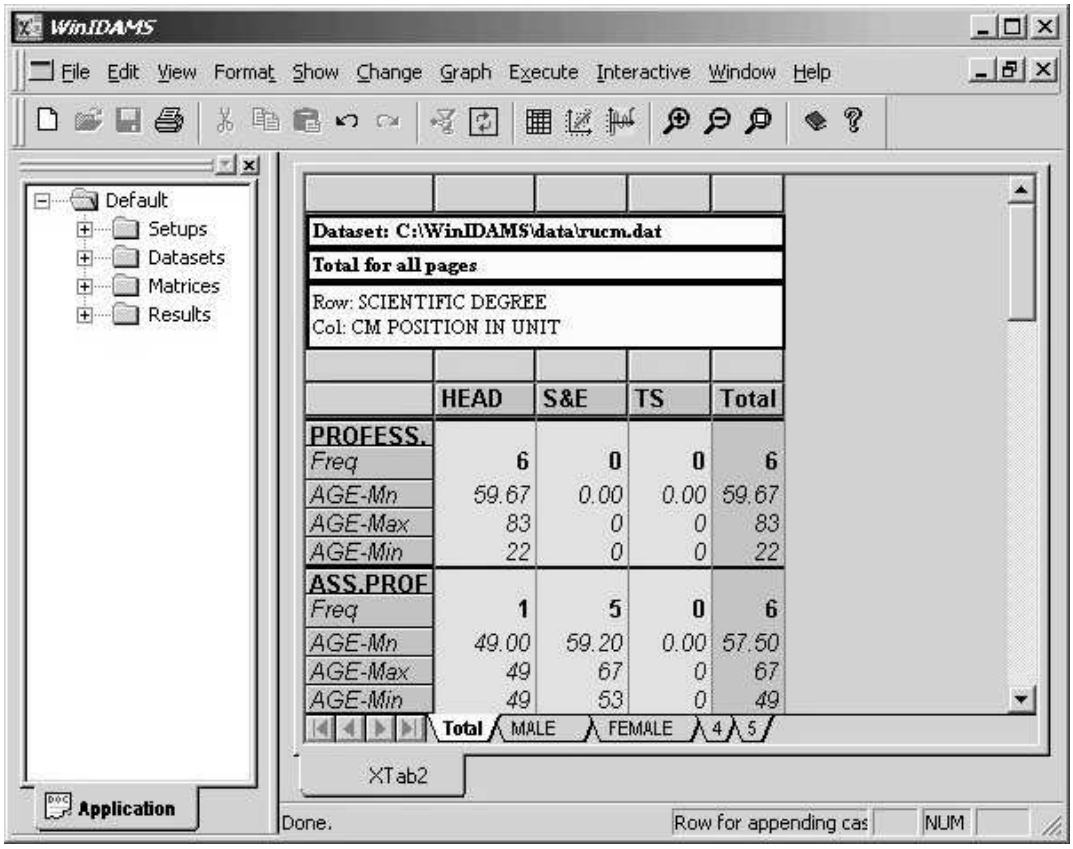
- Mean is marked by default. Mark Max and Min. Then click on OK here and on OK in the Multidimensional Table Definition dialogue. You now see the multidimensional table.



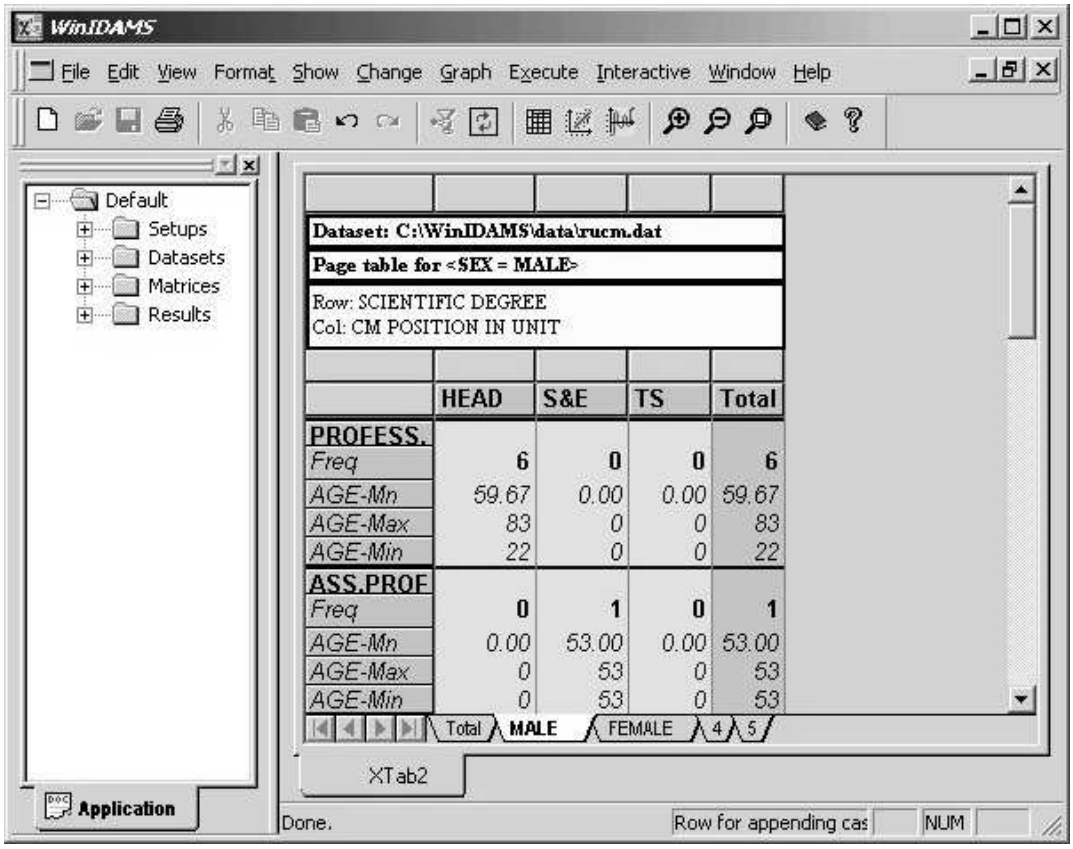
39.6 How to Change a Multidimensional Table

Asking for separate tables. Suppose that now you wish to see a separate table for the men and the women.

- Click on Change/Specification and you get back the dialogue with your previous selection of variables.
- Use the Drag and Drop technique to move the "SEX" variable from the ROW VARIABLES list to the PAGE VARIABLES list and click on OK.
- You see the first view which is the total for all values taken together (men and women). At the bottom of the view you can see three tabs: "Total", "MALE" and "FEMALE". "Total" is the tab of the current view.



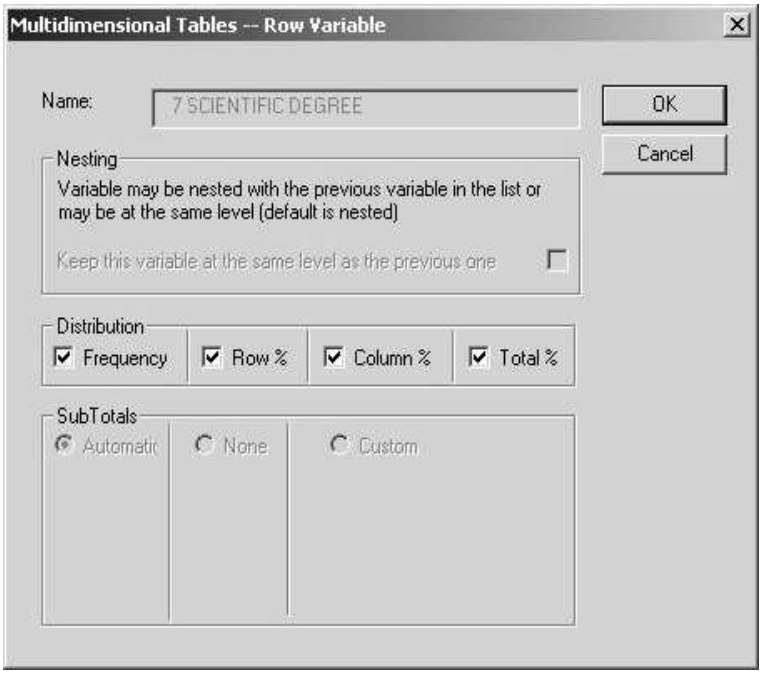
- To see the page for the men, click on tab "MALE".



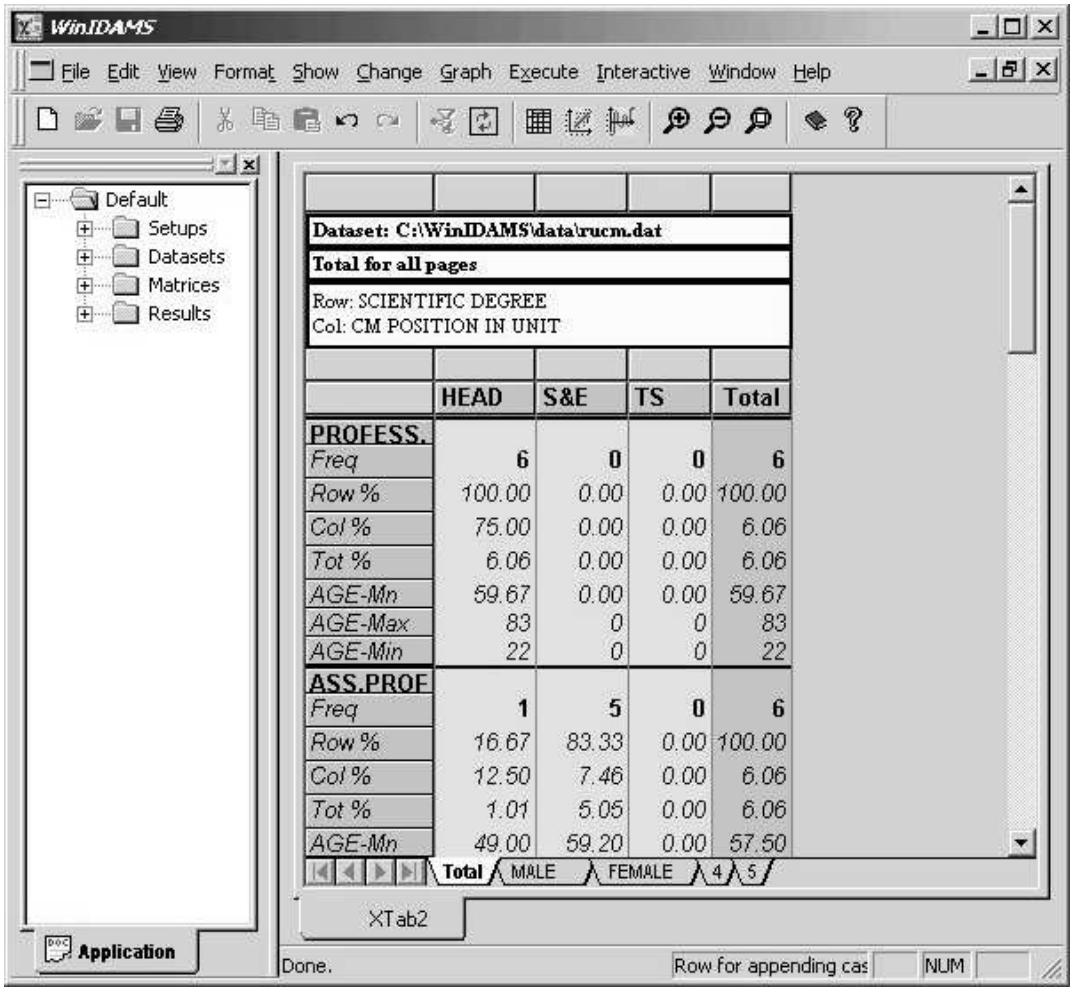
- To see the page for women, click on tab "FEMALE".

Asking for the percentages. While frequencies are displayed by default, any type of percentages must be requested explicitly.

- Click on Change/Specification and you get back the dialogue with your previous selection of variables.
- Double-click on the row variable "SCIENTIFIC DEGREE" and you see a dialogue with boxes for Frequency (marked by default), Row %, Column % and Total %. Mark all the percentage boxes as follow:



- Click on OK for accepting this change and click on OK in the Multidimensional Table Definition dialogue. You see the previous multidimensional table with all percentages.


WinIDAMS
File Edit View Format Show Change Graph Execute Interactive Window Help
Default
Setups
Datasets
Matrices
Results
Application
Dataset: C:\WinIDAMS\data\rucm.dat
Total for all pages
Row: SCIENTIFIC DEGREE
Col: CM POSITION IN UNIT
HEAD S&E TS Total
PROFESS.
Freq 6 0 0 6
Row % 100.00 0.00 0.00 100.00
Col % 75.00 0.00 0.00 6.06
Tot % 6.06 0.00 0.00 6.06
AGE-Mn 59.67 0.00 0.00 59.67
AGE-Max 83 0 0 83
AGE-Min 22 0 0 22
ASS.PROF
Freq 1 5 0 6
Row % 16.67 83.33 0.00 100.00
Col % 12.50 7.46 0.00 6.06
Tot % 1.01 5.05 0.00 6.06
AGE-Mn 49.00 59.20 0.00 57.50
Total MALE FEMALE 4 5
XTab2
Done.
Row for appending cas
NUM

Chapter 40

Graphical Exploration of Data (GraphID)

40.1 Overview

GraphID is a component of WinIDAMS for interactive exploration of data through graphical visualization. It accepts two kinds of input:

- IDAMS datasets where the Dictionary and Data files must have the same name with extensions .dic and .dat respectively,
- IDAMS Matrix files where the extension must be .mat.

Only one dataset or one matrix file can be used at a time, i.e. opening of another file automatically closes the one being used.

40.2 Preparation of Analysis

Selection of data. Use the menu command File/Open or click the toolbar button Open. Then, in the Open dialogue box, choose your file. Setting "Files of type:" to "IDAMS Data File (*.dat)" or to "IDAMS Matrix File (*.mat)" allows for filtering of files displayed.

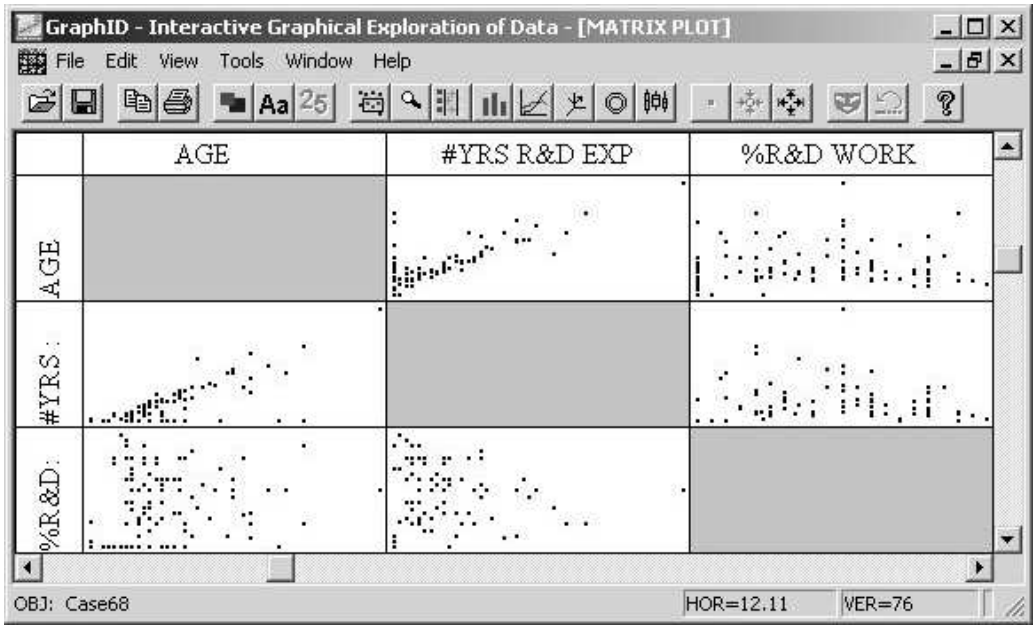
Selection of case identification. If you have selected a dataset, you are asked to specify a case identification which can be a variable or the case sequence number. A numeric or alphabetic variable can be selected from a drop-down list.

Selection of variables. If you have selected a dataset, you are asked to specify the variables which you want to analyse. Numeric variables can be selected from the "Source list" and moved to the "Selected items" area. Moving variables between the lists can be done by clicking the buttons >, < (move only highlighted variables), >>, << (move all variables). Note that alphabetic variables are not available here and that the case identification variable is not allowed for analysis.

Missing data treatment. Two possibilities are proposed: (1) case-wise deletion, when a case is used in analysis only if it has valid data on all selected variables; (2) pair-wise deletion, when a case is used if it has valid data on both variables for each pair of variables separately.

40.3 GraphID Main Window for Analysis of a Dataset

After selection of variables and a click on OK, the GraphID Main window displays the initial matrix of scatter plots with 3 variables and the default properties of the matrix. This display can be manipulated using various options and commands in the menus and/or equivalent toolbar icons.



40.3.1 Menu bar and Toolbar

File

Open	Calls the dialogue box to select a new dataset/matrix file for analysis.
Close	Closes all windows for the current analysis.
Save As	Calls the dialogue box to save the graphical image of the active window in Windows Bitmap format (*.bmp).
Save masked cases	Saves for subsequent use, the sequential number of the cases masked during the session, this following their sequence in the Data file analysed.
Print	Calls the dialogue box to print the contents of the active window.
Print Preview	Displays a print preview of the graphical image in the active window.
Print Setup	Calls the dialogue box for modifying printing and printer options.
Exit	Terminates the GraphID session.

The menu can also contain the list of recently opened files, i.e. files used in previous GraphID sessions.

Edit

The menu has only one command, Copy, to copy the graphic displayed in the active window to the Clipboard.

View

Configuration	Calls the dialogue box for selecting symbols, colours, variables and the number of visible columns and rows in the matrix.
Scales	Displays/hides graph scales for the active zoom window.
Toolbar	Displays/hides toolbar.
Status Bar	Displays/hides status bar.
Info	Displays a window with relevant information about the dataset: number of cases, number of variables, Data file name, etc.
Cell Info	Displays a window with relevant information about the active plot: variable names, their mean values, standard deviations, correlation and regression coefficients.

Brush appearance	Calls the dialogue box to select the symbol and colour for brushed cases.
Font for Scales	Calls the dialogue box to select the font for scales for the active zoom window.
Font for Labels	Calls the dialogue box to select the font for variable names.
Basic Colors	Calls the dialogue box to select colours for the active window: margin colour, grid colour and diagonal cell background.
Save Colors	Saves modification of colours.
Save Fonts	Saves modification of fonts.

Tools

In this menu you can find tools for manipulating the matrix of scatter plots and for calling other graphics provided by GraphID.

Brush	Sets/cancels brush mode.
Zoom	Magnifies the active plot or the brush contents to full window.
Grouping	Calls the dialogue box to specify creation of groups.
Cancel grouping	Cancels grouping.
Histograms	Calls the dialogue box to specify graphics to be shown in the diagonal cells and their properties.
Smoothing	Calls the dialogue box to specify types of regression lines (smoothing lines) and their properties.
3D Scatter Plots	Calls the dialogue box to select variables to be used as axes for 3D-scattering and rotating.
Directed Mode	Sets/cancels directed mode.
Box-Whisker Plots	Calls the dialogue box to select variables and colours for displaying Box-Whiskers plots.
Jittering	Performs jittering of projected cases.
Masking	Mask the cases inside the brush.
Unmasking	Restore step by step masked cases.
Apply saved masking	Mask the cases which were masked and saved in the previous session.
Grouped plot	Calls the dialogue box to select row and column variables for constructing two-dimensional table, and X and Y variables for projecting their scatter plots within the cells of the table.

Window

The menu contains the list of opened windows and Windows commands for arranging them.

Help

WinIDAMS Manual	Provides access to the WinIDAMS Reference Manual.
About GraphID	Displays information about the version and copyright of GraphID and a link for accessing the IDAMS Web page at UNESCO Headquarters.

Toolbar icons

There are 21 buttons in the toolbar providing direct access to the same commands/options as the corresponding menus. They are listed here as they appear from the left to the right.

Open	Brush	Box-Whisker plots
Save	Zoom	Cancel jittering
Copy	Grouping	Decrease jittering level
Print	Histograms	Increase jittering level
Basic colors	Smoothed lines	Mask the cases inside brush
Font for labels	3D scatter plots	Restore step by step masked cases
Font for scales	Directed mode	Information about version of GraphID

40.3.2 Manipulation of the Matrix of Scatter Plots

Configuring the matrix of scatter plots. The current matrix of scatter plots can be changed using the menu command View/Configuration.

Visible: Here you can set the number of columns and rows to be displayed on the screen (they do not need to be equal). Other cells are made visible by scrolling.

Variables: The dialogue box carries two lists of variables: "Source list" and "Selected items". Moving variables between the lists can be done by clicking the buttons >, < (move only highlighted variables), >>, << (move all variables).

Symbols: In this dialogue box, you can select the shape and colour of the symbols that are to be used to represent each group of cases in the plots. If no groups are specified, then all the cases fall in a single group by default and all will be represented by the same symbol (default is a small black rectangle). One can either assign one symbol to one group or collapse groups by assigning the same symbol to two or more groups.

The list of groups is given in the left-hand box. Two other boxes are for selecting colours and symbols. To select a colour or symbol, just click on it. Its image will appear immediately in the button next to the name of the highlighted group.

Directed mode. This option is useful when the order of cases on some column variables is meaningful, e.g. when values of a column variable indicate time intervals. Linking the images sequentially by straight lines can then, for example, help search for cyclical patterns.

To switch to directed plots or come back to scatter plots, press the toolbar button Directed mode or use the menu command Tools/Directed mode.

Masking and Unmasking cases. You can mask cases projected in scatter plots. This feature can be useful, for example, to remove outliers from the graphics.

Masking is available when the brush is active.

To mask cases included in the brush, click the toolbar button Mask. Masked cases are hidden in all the scatter plots. Masking can be repeated several times.

All or part of the masked cases can be unmasked by clicking the toolbar button Restore.

Saving and re-using masked cases. The sequential number of currently masked cases can be saved in a file corresponding to the analysed dataset using the command File/Save masked cases. This masking can be recuperated in subsequent session(s) using the command Tools/Apply saved masking.

Grouping cases. This feature allows you to see how a variable partitions cases into groups in all plots. The variable can be either qualitative or quantitative. In addition to selecting the grouping variable, the user controls the way of grouping (by values, or by intervals and the number of groups).

The dialogue box for creation of groups is activated by clicking the toolbar button Grouping or by using the menu command Tools/Grouping.

Exploration with the brush. The brush is a rectangle which can be (re)sized, moved and zoomed. As it is moved over one scatter plot, the cases inside the brush are highlighted in brush colour and shape on all the other scatter plots.

One of the applications is to determine if a crowding of cases in a scatter plot really represents a cluster in the multidimensional space or whether the crowding is simply a property of the projection. For this purpose, place the brush on a crowding in one scatter plot and observe how these cases are located on other scatter plots. If the same crowding appears on other plots then the crowding may indeed indicate a real cluster. Of course the scatter plots must be chosen so that the distance between cases are of the same order in the different plots.

Another application of the brush is to study the conditional distributions. If the 4 corners of the brush are given by x_{min}, x_{max}, y_{min}, y_{max}, then the cases inside the brush are those that satisfy the conditions:

$$x_{min} < x < x_{max} \quad \text{and} \quad y_{min} < y < y_{max}$$

and the cases satisfying these conditions can be studied in the other scatter plots.

Brush can also be used to mask and search for cases.

To enter brush mode or cancel it, click the toolbar button Brush or use the menu command Tools/Brush.

To place the brush in the desired area, set the cursor at the edge, press the left mouse button, drag and release at the other edge.

To move or resize the brush, set the cursor inside the brush rectangle or on its side, press the left button and drag. Note: To move it quickly to another cell, place the cursor in the desired cell and press the left mouse button.

Zooming. Zooming creates a new window to magnify the selected cell or, in brush mode, to magnify the brush. Such a new zoom window has most of the properties of a matrix of scatter plots with one cell; for example you can use brushing to identify a new set of cases and then zoom again.

If the parent matrix of scatter plots is in brush mode, modification of the brush is reflected immediately in the zoom window; otherwise the zoom window reflects modifications introduced in the selected cell of the parent matrix.

The menu command View/Scales allows you to display scales of variable values for the active zoom window.

Jittering. The function is useful when there are discrete or qualitative variables in the analysed data. In this case, usual matrices of scatter plots may be not very informative since a part or all 2D and 3D projections present 2D or 3D grids and therefore it is impossible to determine visually how many cases coincide in the same grid position and to which groups they belong.

The jittering is a random transformation of data. Data values (x) are modified by adding a "noise" ($a*U$) where U is a uniformly distributed random value from the interval (-0.5, 0.5) and a is a factor to control the jittering level.

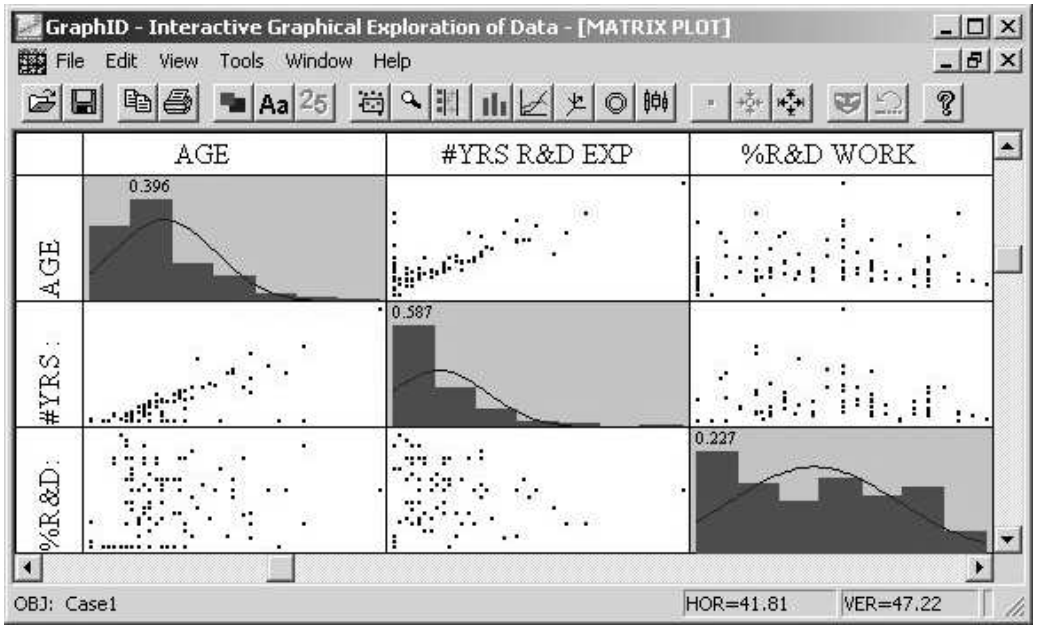
To set the desired jittering level, use the toolbar buttons Decrease jittering level, Increase jittering level and Cancel jittering.

Note that jittering can be performed only in the window of the matrix of scatter plots.

40.3.3 Histograms and Densities

Histograms, normal densities and dot graphics, and three univariate statistics can be displayed in the diagonal cells of the matrix of scatter plots.

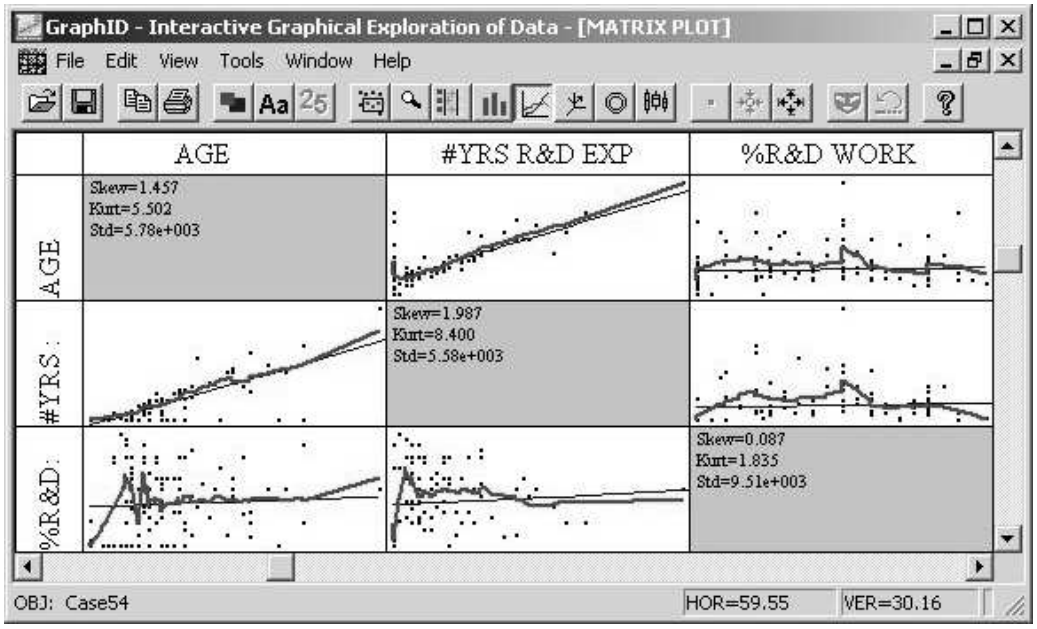
To obtain these, click the toolbar button Histograms or use the menu command Tools/Histograms. In the dialogue box presented you can select the desired graphics, the colour and the number of histogram bars. With the option Statistics, the following statistics are provided: Skewness (Skew), Kurtosis (Kurt) and Standard deviation (Std).



40.3.4 Regression Lines (Smoothed lines)

Up to 4 different regression lines can be displayed on each scatter plot:

MLE (Maximum Likelihood Estimation) linear regression (usual linear regression)
Local linear regression
Local mean
Local median.



Note that these are regression lines of Y versus X, where the X and Y variables are projected respectively on the horizontal and vertical axis.

To get the lines, click the toolbar button Smoothed lines or use the menu command Tools/Smoothing. Then, in the dialogue box select the desired lines, their colour and the smoothing parameter value.

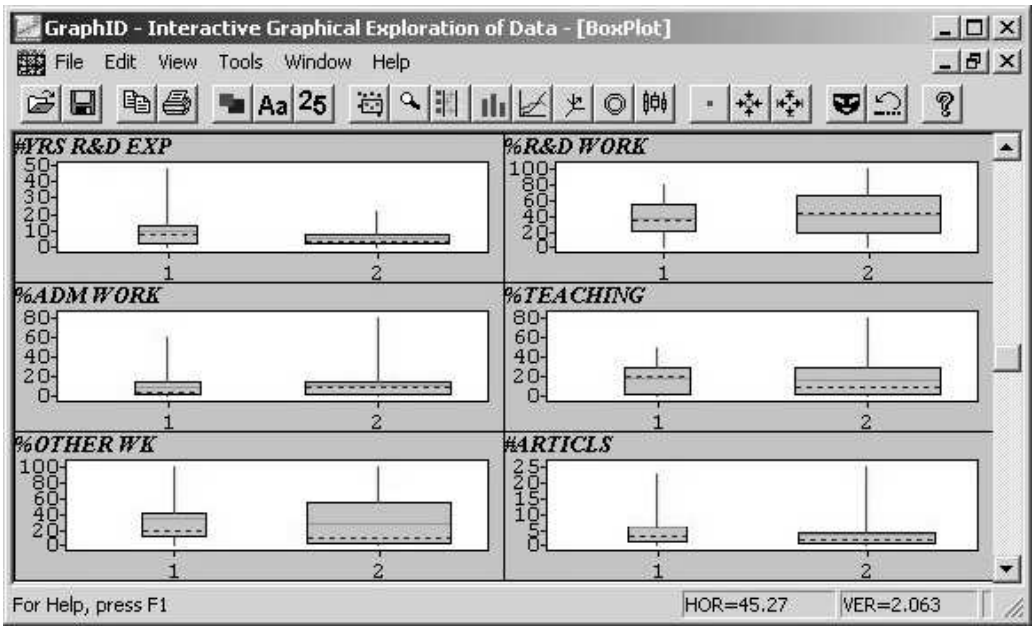
The smoothing parameter is the number of neighbours. It defaults to 7. The value cannot be greater than $n/2$ where n is the number of cases.

40.3.5 Box and Whisker Plots

This feature is especially useful if the cases have been partitioned into groups (see "Grouping cases" above).

Use the menu command Tools/Box-Whisker plots or click the toolbar button "Box-Whisker plots" to get a dialogue box for specifying the number of visible columns and rows as well as colours for the Box and Whisker plots window.

For each selected variable, a graphic image is displayed in the form of a set of boxes, each box corresponding to one group of cases. The base of the box can be set to be proportional to the number of cases in the group, and the upper and lower boundaries show the upper and lower quartiles respectively. The upper and lower ends of vertical lines (whiskers) emerging from the box correspond to the maximum and minimum values of the variable for the group. The lines inside a box are the mean (green line) of the variable in the group and its median (dotted blue line). The left side of a rectangle shows the scale of the variable and its lower margin shows the group numbers.



You may change colours and fonts of the graphics using appropriate buttons in the toolbar. These changes can be saved as new defaults for subsequent windows and sessions.

The **Colors** button allows you to change colours of:

Boxes
Background
Whiskers
Median line
Mean line
Margins.

The **Font** buttons allow you to change fonts for scales and variable names.

Any cell of a Box-Whisker plot can be zoomed. Select the desired cell and click the toolbar button Zoom.

40.3.6 Grouped Plot

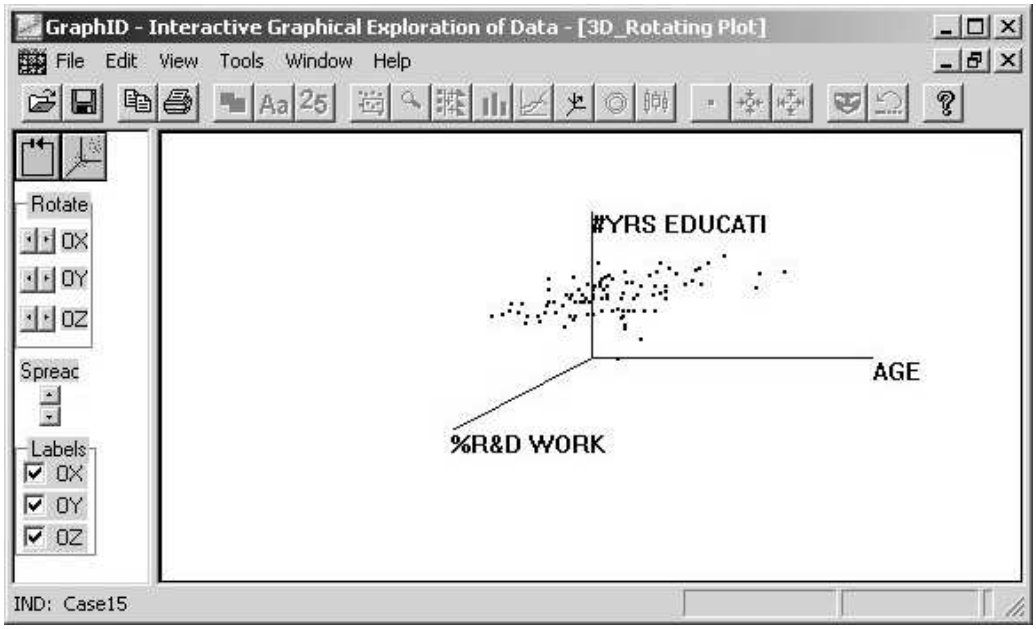
This feature allows projection of a two-dimensional scatter plot within cells of a two-dimensional table, and thus visual analysis in 4 dimensions.

Use the menu command Tools/Grouped plot to get a dialogue box for specifying row and column variables for table construction, and X and Y variables for scatter plots.

You are also requested to select the way of calculating the number of rows and columns. There are two possibilities: they can be equal to the number of distinct variable values or to the user specified number of intervals. Calculated intervals are of the same length.

40.3.7 Three-dimensional Scatter Diagrams and their Rotation

To get a three-dimensional scatter diagram, click the toolbar button 3D scatter plots or use the menu command Tools/3D Scatter Plots. The dialogue box lets you select three variables to be projected along OX, OY and OZ axes. After OK, you get a new window with a three-dimensional scatter diagram for the selected variables. If the parent matrix plot window is in brush mode, the cases included in the brush will be dispayed the same way in this diagram.



You can use the control elements of the dialogue box in the left pane of the window to change the graphical image and to rotate it.

The button in the top left corner can be used to reset the graphics to the start position.

The button in the top right corner can be used to set the center for the cloud of points: either in the gravity center or in the zero point.

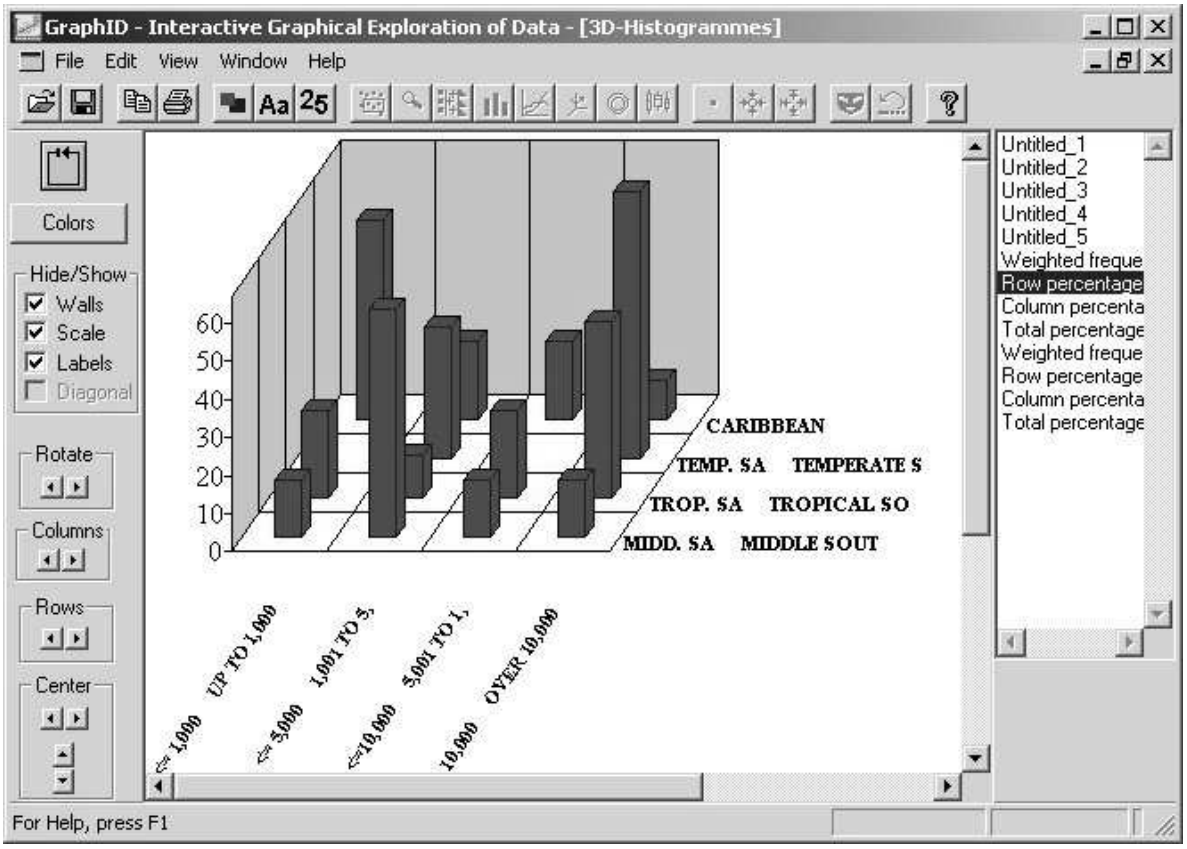
The buttons in the group **Rotate** are used for rotating the scatter diagram around the corresponding axes and the ones in the group **Spread** are used to move points from and towards the center.

The group **Labels** allows you to display or to hide variable names on the corresponding axes.

Finally, the 3D scatter diagram can be projected as three 2D scatter plots by requesting the 2D-view.

40.4 GraphID Window for Analysis of a Matrix

Once the file with matrices has been selected, you can click on Open or double-click on the file name to display a 3D histogram with one bar for each cell of the first matrix in the file. The height of the bar represents the value of the statistic from the matrix transformed using its range, i.e. $h = (s_{val} - s_{min})/(s_{max} - s_{min})$. By default, negative values are shown in blue and positive values in red.



You can select colours for labels (names) and scales, negative and positive values, walls, floor and background. Use the same technique as for Box and Whisker plots.

In the right part of the window you are presented with a list of matrices included in the file. Note that only the first 16 characters of the matrix contents description are displayed. If there is no description, GraphID displays "Untitled_n". You can display the required matrix by clicking its contents description.

The display of the matrix can be manipulated using options and commands in the menu bar items and/or equivalent toolbar icons.

40.4.1 Menu bar and Toolbar

File and Edit

The same commands as the corresponding menus in dataset analysis, except Close, are provided.

View

Toolbar	Displays/hides toolbar.
Status Bar	Displays/hides status bar.
Colors	Calls the dialogue box to select colours for the active window: row/column labels and scales, negative and positive values, walls, floor and background.
Font for Scales	Calls the dialogue box to select the font for scales.
Font for Labels	Calls the dialogue box to select the font for labels.

Window and Help

The same commands as corresponding menus in dataset analysis are available.

Toolbar icons

Buttons are available in the toolbar providing direct access to the same commands/options as the corresponding menus. They are listed here as they appear from the left to the right.

Open
Save
Copy
Print
Colors
Font for Labels
Font for Scales
Information about the version of GraphID.

40.4.2 Manipulation of the Displayed Matrix

Similar to the manipulation of 3D scatter diagrams, you can use the control elements of the dialogue box in the left pane of the window to change the graphical image and to rotate the displayed matrix.

The top button can be used to reset the graphic to the start position.

The **Colors** button lets you change colours of:

Bar (positive values)
Wall
Bar (negative values)
Floor
Background
Labels and scale.

Boxes of the group **Hide/Show** allow you to display or hide walls, scale, labels on the corresponding axes and the diagonal if applicable.

The buttons in the group **Rotate** can be used for rotating the matrix around the vertical axis.

The buttons in the groups **Columns** and **Rows** can be used to change the size of columns and rows respectively.

The buttons in the group **Center** allow you to move the graphic left, right, up and down.

Chapter 41

Time Series Analysis (TimeSID)

41.1 Overview

TimeSID is a component of WinIDAMS for time series analysis. It uses IDAMS datasets as input where the dictionary and data files must have the same name with extensions .dic and .dat respectively.

Only one dataset can be used at a time, i.e. opening of another dataset automatically closes the one being used.

41.2 Preparation of Analysis

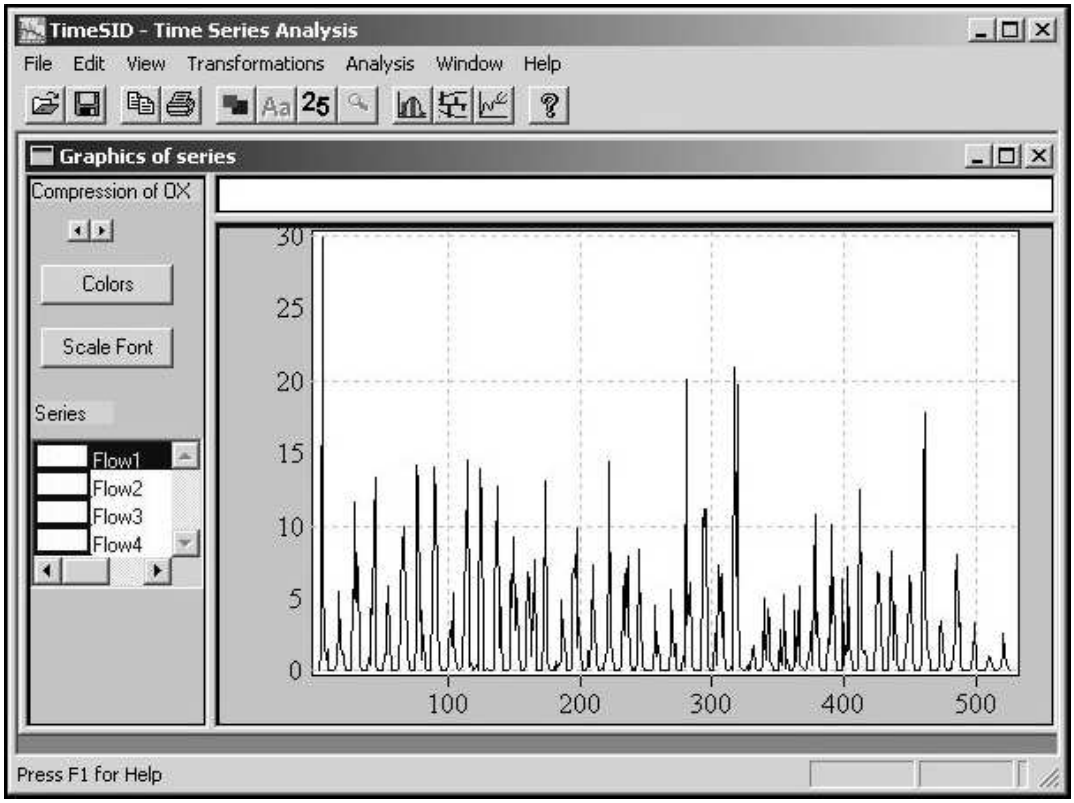
Selection of data. Use the menu command File/Open or click the toolbar button Open. Then, in the Open dialogue box, select your file. Setting "Files of type:" to "IDAMS Data File (*.dat)" displays only IDAMS data files.

Selection of series. You are also asked to specify the series (variables) you want to analyse. Numeric variables can be selected from the "Accessible series" list and moved to the "Selected series" area. Moving variables between the lists can be done by clicking the buttons >, < (move only highlighted variables), >>, << (move all variables). Note that alphabetic variables are not available here.

Missing data treatment. Missing data values are excluded from transformations of series; they are also excluded from calculation of statistics and autocorrelations. For the other analysis, missing data values are replaced by the overall mean.

41.3 TimeSID Main Window

After selection of variables and a click on OK, the TimeSID Main window displays the graphic of the first series from the list of selected series. The series can be manipulated and analysed using various options and commands in the menus and/or equivalent toolbar icons.



41.3.1 Menu bar and Toolbar

File

Open	Calls the dialogue box to select a new dataset for analysis.
Close	Closes all windows for the current analysis.
Save As	Calls the dialogue box to save the contents of the active pane/window. Graphical images are saved in Windows Bitmap format (*.bmp). Data table and tables with statistics are saved in text format.
Print	Calls the dialogue box to print the contents of the active pane/window.
Print Preview	Displays a print preview of the contents of the active pane/window.
Print Setup	Calls the dialogue box for modifying printing and printer options.
Exit	Terminates the TimeSID session.

The menu can also contain the list of recently opened files, i.e. files used in previous TimeSID sessions.

Edit

The menu has only one command, Copy, to copy the contents of the active pane/window to the Clipboard.

View

Toolbar	Displays/hides toolbar.
Status Bar	Displays/hides status bar.
OX Scale	Displays/hides OX scale for the time series.
Font for Scales	Calls the dialogue box to select the font for scales.
Basic Colors	Calls the dialogue box to select colours for the margin and background.

Window

Data Table Calls the window with the data table. Columns of the data table are the analyzed time series (including transformation results).

Besides Data Table, the menu contains the list of opened windows and Windows options for arranging them.

Help

WinIDAMS Manual Provides access to the WinIDAMS Reference Manual.

About TimeSID Displays information about the version and copyright of TimeSID and a link for accessing the IDAMS Web page at UNESCO Headquarters.

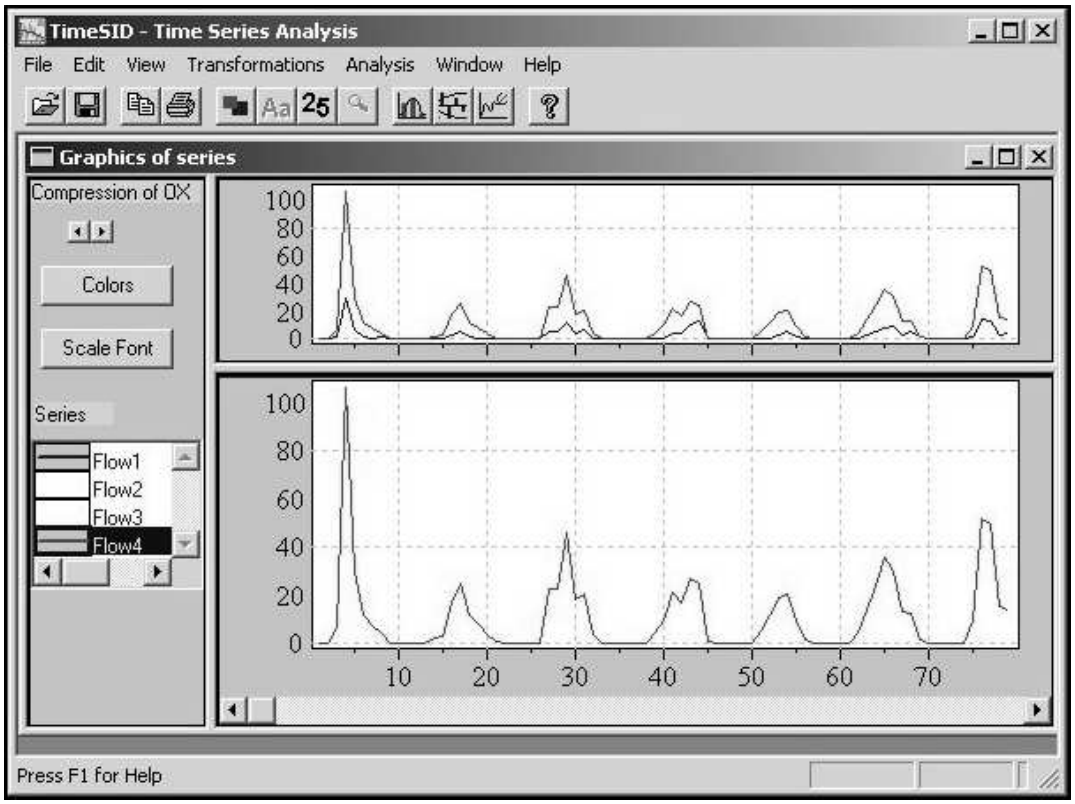
The two other menus, Transformations and Analysis, are described in details in sections "Transformation of Time Series" and "Analysis of Time Series" below.

Toolbar icons

There are 9 active buttons in the toolbar providing direct access to the same commands/options as the corresponding menu items. They are listed here as they appear from the left to the right.

Open	Histograms, basic statistical characteristics
Copy	Auto-, cross-correlation
Print	Auto-regression
Basic colors	Display information about TimeSID
Font for scales	

41.3.2 The Time Series Window



The time series window is divided into 3 panes: the left one is for changing the window properties and for selecting series (variables), the right upper is for displaying several time series and the right lower is for displaying the current series.

Changing the pane appearance. The two panes for displaying time series are synchronized and they can be changed using the controls provided in the left pane. By default, the right upper pane is empty and its size is reduced. The right lower pane displays the current series, keeping scroll bar and scales visible. The size of either pane can be changed using the mouse, and the OX scale can be hidden/displayed using the OX Scale command of the menu View. Moreover, presentation of graphics can be modified as follows:

- *regulation of graphic compression degree* - use the buttons under Compression of OX,
- *colours for background and margins* - use the Colors button or View/Basic Colors command,
- *font for scales* - use the Scale Font button or View/Font for Scales command.

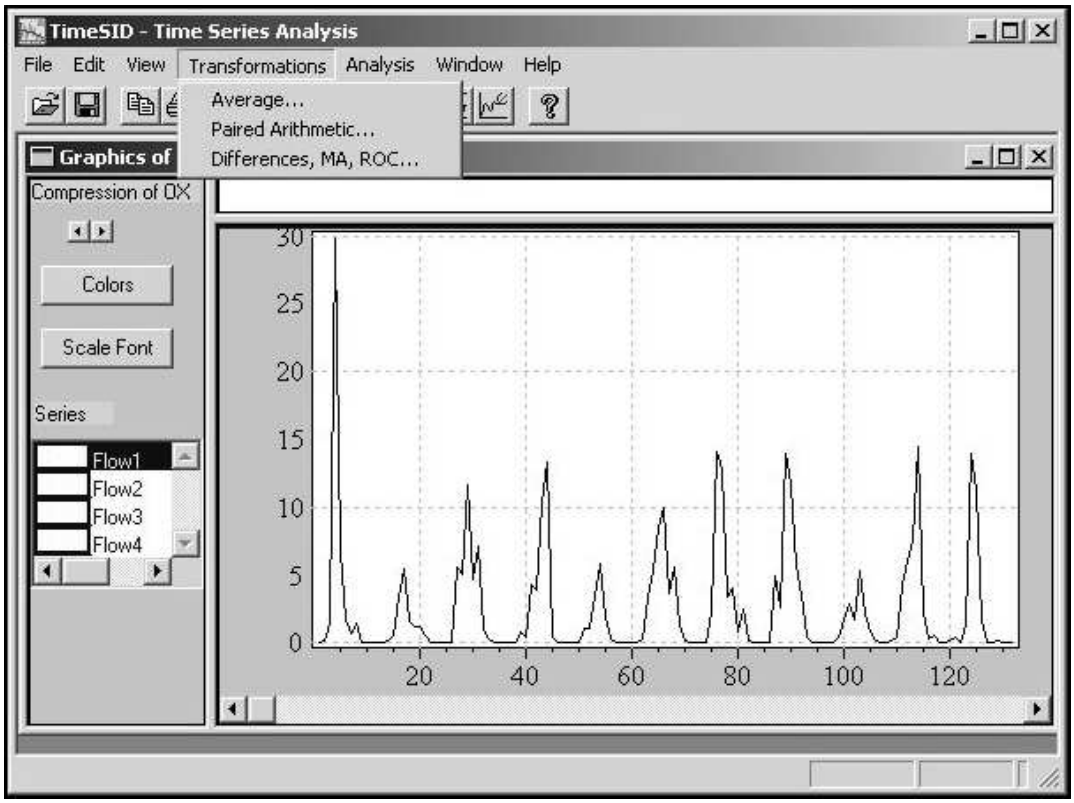
Changing time series name. Select the required time series, click its name with the right mouse button and select the Change name option. The active window presents the name for modification. Note that these modifications are temporary and they are kept only during the current session.

Selecting time series for display. A list of analysed time series is provided in the left pane. By double clicking a variable in the list, you can choose the shape and colour of the line for projection. After OK, the corresponding graphic is displayed in the upper pane. This operation can be repeated for different variables and thus you can get several graphics displayed simultaneously in the upper pane. The right lower pane always displays the current series.

Deleting time series from analysis. Select the required time series, click its name with the right mouse button and select the Delete series option.

41.4 Transformation of Time Series

Time series data can be transformed by calculating differences, smoothing, trend suppression, using a number of functions, etc. The menu **Transformations** contains commands for creating new time series based on values of selected series. Note that variables displayed for selection are renumbered sequentially starting from zero (0).



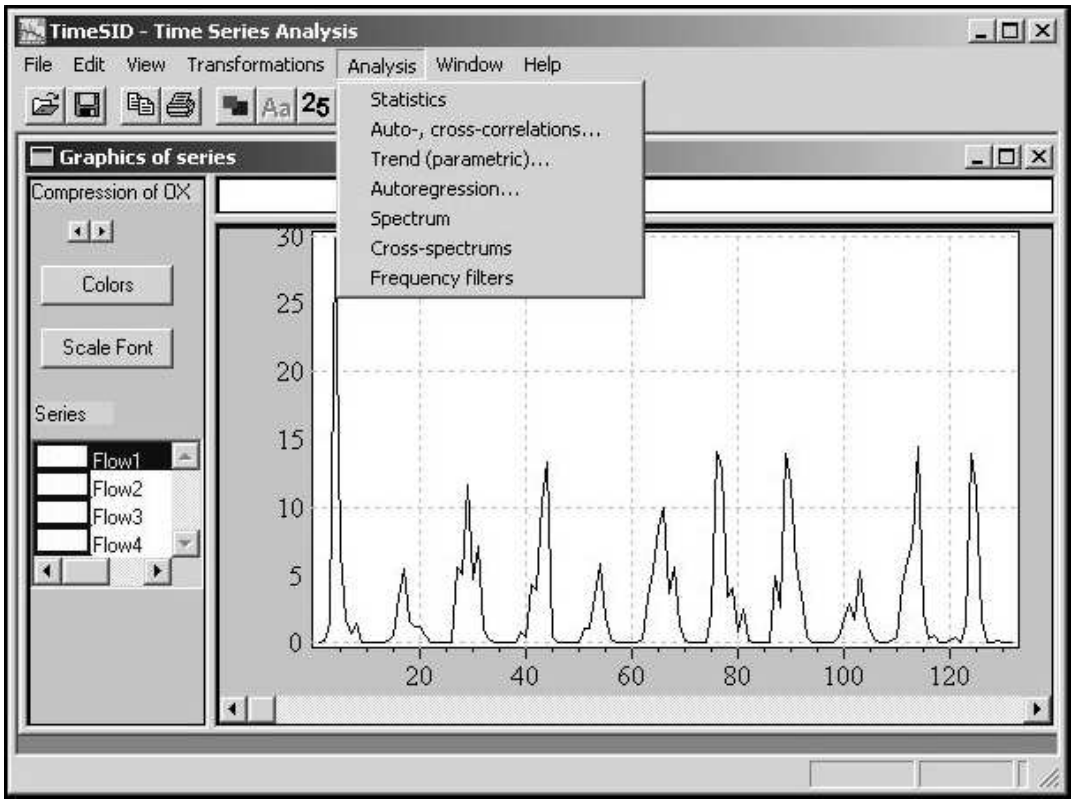
Average creates a new time series as an average of the specified series. Series to be taken for calculation are selected in the dialogue box "Selection of series" (see section "Preparation of Analysis").

Paired arithmetic creates a set of time series by performing arithmetic operations on pairs of time series specified in the dialogue box (each series specified in the first argument list with the second argument).

Differences, MA, ROC creates a set of time series based on transformations (sequential differences, uncentered moving average, rate of change) of the series specified in the dialogue box. Parameters specific for each transformation as well as the type of ROC transformation are set in the same dialogue box.

41.5 Analysis of Time Series

Analysis features are activated through commands in the menu **Analysis.**



Statistics creates the table with mean, standard deviation, minimum and maximum values as well as the table with statistics for testing the hypothesis "randomness versus trend" for the selected time series. It also displays a histogram for this series.

Auto-, cross-correlations creates a new window with a set of cells containing graphs of auto- and cross-correlations for the set of specified time series.

Trend (parametric) creates a new time series as the estimation of a parametric trend model for the specified time series. The trend model and the series are selected in a dialogue box.

Autoregression estimates the parameters of an auto-regression model for short-term prediction for the specified time series.

Spectrum (spectral analysis) creates a table of spectrum values (frequency, period, density), graph of spectrum estimation, and for DFT spectrum, graph of deviations of the cumulative spectrum from the cumulative "white-noise" spectrum. It can use the fast discrete Fourier transformation (DFT) and/or maximal entropy (MENT) method for the spectrum density estimation. In the DFT procedure, two windows are used to get the improved estimation of spectral density: Welch data window in the time domain and a polynomial smoothing in the frequency domain.

Cross-spectrum analyses a pair of stationary time series. It provides the values of cross-spectrum power, phase and coherency function as well as their plots. Cross-spectrum is estimated using the Parzen smoothing window.

Frequency filters procedure decomposes a time series into frequency components. It creates a new series by applying one of the following filters: low frequency, high frequency, band-pass or band-cut. For low or high frequency filter, its frequency bound is equal to the value of the *Frequency* parameter. For band-pass or band-cut filter, the frequency bounds are determined by the interval *(Frequency - Window width, Frequency + Window width)*. An option Detrend allows to detrend the time series before filtering (the trend component is added to the filtering results).

References

Farnum, N.R., Stanton, L.W., *Quantitative Forecasting Methods*, PWS-KENT Publishing Company, Boston, 1989.

Kendall, M.G., Stuart, A., *The Advanced Theory of Statistics*, Volume 3 - Design and Analysis, and time series, Second edition, Griffin, London, 1968.

Marple Jr, S.L., *Digital Spectral Analysis with Applications*, Prentice-Hall, Inc., 1987.

Part VI

Statistical Formulas and Bibliographical References

Chapter 42

Cluster Analysis

Notation

x	=	values of variables
h, i, j, l	=	subscripts for objects
f, g	=	subscripts for variables
p	=	number of variables
c	=	subscript for cluster
k	=	number of clusters
N_j	=	number of objects in cluster j
N	=	total number of cases.

42.1 Univariate Statistics

If the input is an IDAMS dataset, the following statistics are calculated for all variables used in the analysis:

a) ***Mean.***

$$\overline{x}_f = \frac{\sum_i x_{if}}{N}$$

b) ***Mean absolute deviation.***

$$s_f = \frac{\sum_i |x_{if} - \overline{x}_f|}{N}$$

42.2 Standardized Measurements

In the same situation, the program can compute standardized measurements, also called z-scores, given by:

$$z_{if} = \frac{x_{if} - \overline{x}_f}{s_f}$$

for each case i and each variable f using the mean value and the mean absolute deviation of the variable f (see section 1 above).

42.3 Dissimilarity Matrix Computed From an IDAMS Dataset

The elements d_{ij} of a dissimilarity matrix measure the degree of dissimilarity between cases i and j. The d_{ij} are calculated directly from the raw data, or from the z-scores if the variables are requested to be standardized. One of two distances can be chosen: Euclidean or city block.

a) ***Euclidean distance.***

$$d_{ij} = \sqrt{\sum_{f=1}^{p}(x_{if} - x_{jf})^2}$$

b) ***City block distance.***

$$d_{ij} = \sum_{f=1}^{p}|x_{if} - x_{jf}|$$

42.4 Dissimilarity Matrix Computed From a Similarity Matrix

If the input consists of a similarity matrix with elements s_{ij}, the elements d_{ij} of the dissimilarity matrix are calculated as follows:

$$d_{ij} = 1 - s_{ij}$$

42.5 Dissimilarity Matrix Computed From a Correlation Matrix

If the input consists of a correlation matrix with elements r_{ij}, the elements d_{ij} of the dissimilarity matrix are calculated using one of two formulas: SIGN or ABSOLUTE.

When using the SIGN formula, variables with a high positive correlation receive a dissimilarity coefficient close to zero, whereas variables with a strong negative correlation will be considered very dissimilar.

$$d_{ij} = (1 - r_{ij})/2$$

When using the ABSOLUTE formula, variables with a high positive or strong negative correlation will be assigned a small dissimilarity.

$$d_{ij} = 1 - |r_{ij}|$$

42.6 Partitioning Around Medoids (PAM)

The algorithm searches for k representative objects (medoids) which are centrally located in the clusters they define. The representative object of a cluster, the medoid, is the object for which the average dissimilarity to all the objects in the cluster is minimal. Actually, the PAM algorithm minimizes the sum of dissimilarities instead of the average dissimilarity.

The selection of k medoids is performed in two phases. In the first phase, an initial clustering is obtained by the successive selection of representative objects until k objects have been found. The first object is the one for which the sum of the dissimilarities to all the other objects is as small as possible. (This is a kind of "multivariate median" of the N objects, hence the term "medoid".) Subsequently, at each step, PAM selects the object which decreases the objective function (sum of dissimilarities) as much as possible. In the second phase, an attempt is made to improve the set of representative objects. This is done by considering all pairs of objects (i, h) for which object i has been selected and object h has not, checking whether selecting h and deselecting i reduces the objective function. In each step, the most economical swap is carried out.

a) ***Final average distance (dissimilarity).*** This is the PAM objective function, which can be seen as a measure of "goodness" of the final clustering.

$$\text{Final average distance} = \frac{\sum_{i=1}^{N} d_{i,m(i)}}{N}$$

where $m(i)$ is the representative object (medoid) closest to object i.

b) ***Isolated clusters.*** There are two types of isolated clusters: L-clusters and L^*-clusters.

Cluster $\mathcal{C}$ is an L-cluster if for each object i belonging to $\mathcal{C}$

$$\max_{j \in \mathcal{C}} d_{ij} < \min_{h \notin \mathcal{C}} d_{ih}$$

Cluster $\mathcal{C}$ is an L^*-cluster if

$$\max_{i,j \in \mathcal{C}} d_{ij} < \min_{l \in \mathcal{C}, h \notin \mathcal{C}} d_{lh}$$

c) ***Diameter of a cluster.*** The diameter of the cluster $\mathcal{C}$ is defined as the biggest dissimilarity between objects belonging to $\mathcal{C}$:

$$\text{Diameter}_{\mathcal{C}} = \max_{i,j \in \mathcal{C}} d_{ij}$$

d) ***Separation of a cluster.*** The separation of the cluster $\mathcal{C}$ is defined as the smallest dissimilarity between two objects, one of which belongs to cluster $\mathcal{C}$ and the other does not.

$$\text{Separation}_{\mathcal{C}} = \min_{l \in \mathcal{C}, h \notin \mathcal{C}} d_{lh}$$

e) ***Average distance to a medoid.*** If j is the medoid of cluster $\mathcal{C}$, the average distance of all objects of $\mathcal{C}$ to j is calculated as follows:

$$\text{Average distance}_j = \frac{\sum_{i \in \mathcal{C}} d_{ij}}{N_j}$$

f) ***Maximum distance to a medoid.*** If object j is the medoid of cluster $\mathcal{C}$, the maximum distance of all objects of $\mathcal{C}$ to j is calculated as follows:

$$\text{Maximum distance}_j = \max_{i \in \mathcal{C}} d_{ij}$$

g) ***Silhouettes of clusters.*** Each cluster is represented by a silhouette (Rousseeuw 1987), showing which objects lie well within the cluster and which ones merely hold an intermediate position. For each object, the following information is provided:

- the number of the cluster to which it belongs (CLU),
- the number of the neighbor cluster (NEIG),
- the value s_i (denoted as S(I) in the printed output),
- the three-character identifier of object i,
- a line, the length of which is proportional to s_i.

For each object i the value s_i is calculated as follows:

$$s_i = \frac{b_i - a_i}{\max(a_i,\ b_i)}$$

where a_i is the average dissimilarity of object i to all other objects of the cluster $\mathcal{A}$ to which i belongs and b_i is the average dissimilarity of object i to all objects of the closest cluster $\mathcal{B}$ (neighbor of object i). Note that the neighbor cluster is like the second-best choice for object i. When cluster $\mathcal{A}$ contains only one object i, the s_i is set to zero ($s_i = 0$).

h) ***Average silhouette width of a cluster.*** It is the average of s_i for all objects i in a cluster.

i) ***Average silhouette width.*** It is the average of s_i for all objects i in the data, i.e. average silhouette width for k clusters. This can be used to select the "best" number of clusters, by choosing that k yielding the highest average of s_i.

Another coefficient, SC, called the SILHOUETTE COEFFICIENT, can be calculated manually as the maximum average silhouette width over all k for which the silhouettes can be constructed. This coefficient is a dimensionless measure of the amount of clustering structure that has been discovered by the classification algorithm.

$$\mathrm{SC} = \max_k \overline{s}_k$$

Rousseeuw (1987) proposed the following interpretation of the SC coefficient:

$0.71 - 1.00$	A strong structure has been found.
$0.51 - 0.70$	A reasonable structure has been found.
$0.26 - 0.50$	The structure is weak and could be artificial; please try additional methods on this data.
≤ 0.25	No substantial structure has been found.

42.7 Clustering LARge Applications (CLARA)

Similarly to PAM, the CLARA method is also based on the search for k representative objects. But the CLARA algorithm is designed especially for analyzing large data sets. Consequently, the input to CLARA has to be an IDAMS dataset.

Internally, CLARA carries out two steps. First a sample is drawn from the set of objects (cases), and divided into k clusters using the same algorithm as in PAM. Then, each object not belonging to the sample is assigned to the nearest among the k representative objects. The quality of this clustering is defined as the average distance between each object and its representative object. Five such samples are drawn and clustered in turn, and the one is selected for which the lowest average distance was obtained.

The retained clustering of the entire data set is then analyzed further. The final average distance, the average and maximum distances to each medoid are calculated the same way as in PAM (for all objects, and not only for those in the selected sample). Silhouettes of clusters and related statistics are also calculated the same way as in PAM, but only for objects in the selected sample (since the entire silhouette plot would be too large to print).

42.8 Fuzzy Analysis (FANNY)

Fuzzy clustering is a generalization of partitioning, which can be applied to the same type of data as the method PAM, but the algorithm is of a different nature. Instead of assigning an object to one particular cluster, FANNY gives its degree of belonging (membership coefficient) to each cluster, and thus provides much more detailed information on the structure of the data.

a) ***Objective function.*** The fuzzy clustering technique used in FANNY aims to minimize the objective function

$$\text{Objective function} = \sum_{c=1}^{k} \frac{\sum_i \sum_j u_{ic}^2 \, u_{jc}^2 \, d_{ij}}{2 \sum_j u_{jc}^2}$$

where u_{ic} and u_{jc} are membership functions which are subject to the constraints

$$u_{ic} \geq 0 \qquad \text{for} \quad i = 1, 2, \ldots, N;\ c = 1, 2, \ldots, k$$

$$\sum_c u_{ic} = 1 \quad \text{for} \quad i = 1, 2, \ldots, N$$

The algorithm minimizing this objective function is iterative, and stops when the function converges.

b) ***Fuzzy clustering (memberships).*** These are the membership values (membership coefficients u_{ic}) which provide the smallest value of the objective function. They indicate, for each object i, how strongly it belongs to cluster c. Note that the sum of membership coefficients equals 1 for each object.

c) ***Partition coefficient of Dunn.*** This coefficient, F_k, measures how "hard" a fuzzy clustering is. It varies from the minimum of $1/k$ for a completely fuzzy clustering (where all $u_{ic} = 1/k$) up to 1 for an entirely hard clustering (where all $u_{ic} = 0$ or 1).

$$F_k = \sum_{i=1}^{N} \sum_{c=1}^{k} u_{ic}^2 \,/\, N$$

d) ***Normalized partition coefficient of Dunn.*** The normalized version of the partition coefficient of Dunn always varies from 0 to 1, whatever value of k was chosen.

$$F'_k = \frac{F_k \;-\; (1/k)}{1 \;-\; (1/k)} = \frac{kF_k \;-\; 1}{k \;-\; 1}$$

e) ***Closest hard clustering.*** This partition (= "hard" clustering) is obtained by assigning each object to the cluster in which it has the largest membership coefficient. Silhouettes of clusters and related statistics are calculated the same way as in PAM.

42.9 AGglomerative NESting (AGNES)

This method can be applied to the same type of data as the methods PAM and FANNY. However, it is no longer necessary to specify the number of clusters required. The algorithm constructs a tree-like hierarchy which implicitly contains all values of k, starting with N clusters and proceeding by successive fusions until a single cluster is obtained with all the objects.

In the first step, the two closest objects (i.e. with smallest inter-object dissimilarity) are joined to constitute a cluster with two objects, whereas the other clusters have only one member. In each succeeding step, the two closest clusters (with smallest inter-object dissimilarity) are merged.

a) ***Dissimilarity between two clusters.*** In the AGNES algorithm, the group average method of Sokal and Michener (sometimes called "unweighted pair-group average method") is used to measure dissimilarities between clusters.

Let $\mathcal{R}$ and $\mathcal{Q}$ denote two clusters and $|\mathcal{R}|$ and $|\mathcal{Q}|$ denote their number of objects. The dissimilarity $d(\mathcal{R}, \mathcal{Q})$ between clusters $\mathcal{R}$ and $\mathcal{Q}$ is defined as the average of all dissimilarities d_{ij}, where i is any object of $\mathcal{R}$ and j is any object of $\mathcal{Q}$.

$$d(\mathcal{R}, \mathcal{Q}) = \frac{1}{|\mathcal{R}| \; |\mathcal{Q}|} \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{Q}} d_{ij}$$

b) ***Final ordering of objects and dissimilarities between them.*** In the first line, the objects are listed in the order they will appear in the graphical representation of results. In the second line, the dissimilarities between joining clusters are printed. Note that the number of dissimilarities printed is one less than the number of objects N, because there are $N - 1$ fusions.

c) ***Dissimilarity banner.*** It is a graphical presentation of the results. A banner consists of stars and stripes. The stars indicate links and the stripes are repetitions of identifiers of objects. A banner is always read from left to right. Each line with stars starts at the dissimilarity between the clusters being merged. There are fixed scales above and below the banner, going from 0.00 (dissimilarity 0) to 1.00 (largest dissimilarity encountered). The actual highest dissimilarity (corresponding to 1.00 in the banner) is provided just below the banner.

d) ***Agglomerative coefficient.*** The average width of the banner is called the agglomerative coefficient (AC). It describes the strength of the clustering structure that has been found.

$$AC = \frac{1}{N}\sum_i l_i$$

where l_i is the length of the line containing the identifier of object i.

42.10 DIvisive ANAlysis (DIANA)

The method DIANA can be used for the same type of data as the method AGNES. Although AGNES and DIANA produce similar output, DIANA constructs its hierarchy in the opposite direction, starting with one large cluster containing all objects. At each step, it splits up a cluster into two smaller ones, until all clusters contain only a single element. This means that for N objects, the hierarchy is built in $N-1$ steps.

In the first step, the data are split into two clusters by making use of dissimilarities. In each subsequent step, the cluster with the largest diameter (see 6.c above) is split in the same way. After $N-1$ divisive steps, all objects are apart.

a) ***Average dissimilarity to all other objects.*** Let $\mathcal{A}$ denote a cluster and $|\mathcal{A}|$ denote its number of objects. The average dissimilarity between object i and all other objects in cluster $\mathcal{A}$ is defined as in 6.g above.

$$d_i = \frac{1}{|\mathcal{A}|-1}\sum_{j\in\mathcal{A}, j\neq i} d_{ij}$$

b) ***Final ordering of objects and diameters of clusters.*** In the first line, the objects are listed in the order they will appear in the graphical representation. The diameters of clusters are printed below that. These two sequences of numbers together characterize the whole hierarchy. The largest diameter indicates the level at which the whole data set is split. The objects on the left side of this value constitute one cluster, and the objects on the right side constitute another one. The second largest diameter indicates the second split, etc.

c) ***Dissimilarity banner.*** As for the AGNES method, it is a graphical presentation of the results. It also consists of lines with stars, and the stripes which repeat the identifiers of objects. The banner is read from left to right but the fixed scales above and below the banner now go from 1.00 (corresponding to the diameter of the entire data set) to 0.00 (corresponding to the diameter of singletons). Each line with stars ends at the diameter at which the cluster is split. The actual diameter of the data set (corresponding to 1.00 in the banner) is provided just below the banner.

d) ***Divisive coefficient.*** The average width of the banner is called the divisive coefficient (DC). It describes the strength of the clustering structure found.

$$DC = \frac{1}{N}\sum_i l_i$$

where l_i is the length of the line containing the identifier of object i.

42.11 MONothetic Analysis (MONA)

The method MONA is intended for data consisting exclusively of binary (dichotomic) variables (which take only two values, so that $x_{if} = 0$ or $x_{if} = 1$). Although the algorithm is of the hierarchical divisive type, it does not use dissimilarities between objects, and therefore a matrix of dissimilarities is not computed. The division into clusters uses the variables directly.

At each step, one of the variables (say, f) is used to split the data by separating the objects i for which $x_{if} = 1$ from those for which $x_{if} = 0$. In the next step, each cluster obtained in the previous step is split

further, using values (0 and 1) of one of the remaining variables (different variables may be used in different clusters). The process is continued until each cluster either contains only one object, or the remaining variables cannot split it.

For each split, the variable most strongly associated with the other variables is chosen.

a) ***Association between two variables.*** The measure of association between two variables f and g is defined as follows:

$$A_{fg} = |a_{fg}d_{fg} - b_{fg}c_{fg}|$$

where a_{fg} is the number of objects i with $x_{if} = x_{ig} = 0$, d_{fg} is the number of objects with $x_{if} = x_{ig} = 1$, b_{fg} is the number of objects with $x_{if} = 0$ and $x_{ig} = 1$, and c_{fg} is the number of objects with $x_{if} = 1$ and $x_{ig} = 0$.

The measure A_{fg} expresses whether the variables f and g provide similar divisions of the set of objects, and can be considered as a kind of similarity between variables.

In order to select the variable most strongly associated with the other variables, the total measure A_f is calculated for each variable f as follows:

$$A_f = \sum_{g \neq f} A_{fg}$$

b) ***Final ordering of objects.*** The objects are listed in the order they appear in the separation plot (banner). The separation steps and the variables used for separation are printed under object identifiers.

c) ***Separation plot (banner).*** This graphical presentation is quite similar to the banner printed by DIANA. The length of a row of stars is now proportional to the step number at which separation was carried out. Rows of object identifiers correspond to objects. A row of identifiers which does not continue to the right-hand side of the banner signals an object that became a singleton cluster at the corresponding step. Rows of identifiers plotted between two rows of stars indicate objects belonging to a cluster which cannot be separated.

42.12 References

Kaufman, L., and Rousseeuw, P.J., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, Inc., New York, 1990.

Rousseeuw, P.J., Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis, *Journal of Computational and Applied Mathematics*, 20, 1987.

Chapter 43

Configuration Analysis

Notation

Let $\mathcal{A}_{(n,t)}$ be a rectangular matrix of n variables(rows) and t dimensions(columns). A variable or point a has t coordinates, each one corresponding to one dimension.

a_{is}	=	element of the matrix $\mathcal{A}$ in the i^{th} row and the s^{th} column
i, j	=	subscripts for variables(rows)
n	=	number of variables
s, l, m	=	subscripts for dimensions(columns)
t	=	number of dimensions.

43.1 Centered Configuration

The variables are centered within each dimension by subtracting the mean of each column from each element in the column.

$$\text{Centered } a_{is} = a_{is} - \frac{\sum_i a_{is}}{n}$$

After application of this formula, the mean of the coordinates of the n variables is zero for each dimension.

43.2 Normalized Configuration

The sum of squares of all the elements of the matrix $\mathcal{A}$ divided by the number of variables n gives the mean of second moments of the variables. Each element of the matrix is normalized by the square root of this value (see denominator below).

$$\text{Normalized } a_{is} = \frac{a_{is}}{\sqrt{\sum_i \sum_s a_{is}^2 / n}}$$

After this normalization, the sum of squares of the a_{is} elements is equal to n.

43.3 Solution with Principal Axes

The configuration is rotated so that successive dimensions account for maximum possible variance. Let $\mathcal{A}$ be the configuration to be rotated and $\mathcal{B}$ be the configuration in its principal axis form.

Calculation of matrix $\mathcal{B}$:

The symmetric matrix $\mathcal{A}'\mathcal{A}$ of dimensions (t,t) is computed first. Then the eigenvectors, $\mathcal{T}$, of $\mathcal{A}'\mathcal{A}$ are determined using Jacobi's diagonalization method.

The matrix $\mathcal{A}$ is transformed into a matrix $\mathcal{B}$ of b_{is} elements, such that $\mathcal{B} = \mathcal{A}\mathcal{T}$, $\mathcal{B}$ having n lines and t columns like the matrix $\mathcal{A}$.

43.4 Matrix of Scalar Products

$$\mathrm{SP}_{ij} = \sum_s a_{is}\, a_{js}$$

The matrix SP of dimensions (n,n) is a square and symmetric matrix of scalar products of variables. The scalar product of a variable by itself is its second moment. If each variable is centered and normalized (mean $= 0$, standard deviation $= 1$), the matrix SP becomes a correlation matrix.

43.5 Matrix of Interpoint Distances

$$\mathrm{DIST}_{ij} = \sqrt{\sum_s (a_{is} - a_{js})^2}$$

DIST is a square and symmetric matrix of Euclidean distances between variables.

43.6 Rotated Configuration

The rotation can be performed only on two dimensions at a time. It belongs to the user to select the dimensions, e.g. 2 and 5 (column 2 and column 5) and the angle ϕ of rotation in terms of degrees.

New coordinates are calculated as follows:

$$\begin{aligned} a'_{il} &= a_{il}\cos\phi + a_{im}\sin\phi \\ a'_{im} &= -a_{il}\sin\phi + a_{im}\cos\phi \end{aligned}$$

The calculation is performed for each value of i, and as many times as that there are variables.

In the matrix $\mathcal{A}$, the columns l and m become the vectors of the new coordinates calculated as indicated above.

43.7 Translated Configuration

The translation can be performed only on one single dimension(one column) at a time. The user specifies the constant T to be added to each element of the dimension, and the column l it applies to.

For all the coordinates of l (n coordinates since n variables):

$$a'_{il} = a_{il} + T$$

43.8 Varimax Rotation

(a) The elements a_{is} of $\mathcal{A}$ are normalized by the square root of the communalities corresponding to each variable, and one defines

$$b_{is} = \frac{a_{is}}{\sqrt{\sum_s a_{is}^2}}$$

(b) Having constructed $\mathcal{B} = (b_{is})$, one looks for the best projection axes for the variables, after equalization of their inertia. The maximization of the function V_c is performed through successive rotations of two dimensions at a time, until convergence is reached.

$$V_c = \sum_s \frac{n \sum_i b_{is}^4 - \left(\sum_i b_{is}\right)^2}{n^2}$$

The result matrix $\mathcal{B}$ of b_{is} elements has the same number of lines and columns as the initial matrix $\mathcal{A}$.

43.9 Sorted Configuration

This is the final configuration printed in a different format. Each dimension is printed as a row, with elements for the dimension in ascending order.

43.10 References

Greenstadt, J., The determination of the characteristic roots of a matrix by the Jacobi method, *Mathematical Methods for Digital Computers*, eds. A. Ralston and H.S. Wilf, Wiley, New York, 1960.

Herman, H.H., *Modern Factor Analysis*, University of Chicago Press, Chicago, 1967.

Kaiser, H.F., Computer program for varimax rotation in factor analysis, *Educational and Psychological Measurement*, 3, 1959.

Chapter 44

Discriminant Analysis

Notation

x	=	values of variables
k	=	subscript for case
i, j	=	subscripts for variables
g	=	superscript for group
q	=	subscript for step
p	=	number of variables
w	=	value of the weight
x_k^g	=	p elements' vector corresponding to the case k in the group g
y_q^g	=	vector with mean values of variables selected in the step q for the group g
N^g	=	number of cases in the group g
W^g	=	total sum of weights for the group g
I_q	=	subset of indices for variables selected in the step q.

44.1 Univariate Statistics

These statistics, weighted if the weight is specified, are calculated for each group and for each analysis variable, using the basic sample. The mean is calculated also for the whole basic sample (total mean).

a) ***Mean.***

$$\overline{x}_i^g = \frac{\sum_{k=1}^{N^g} w_k^g \, x_{ki}^g}{W^g}$$

Note: the total mean is calculated using the analogous formula.

b) ***Standard deviation.***

$$s_i^g = \sqrt{\frac{\sum_{k=1}^{N^g} w_k^g \left(x_{ki}^g\right)^2}{W^g} - \left(\overline{x}_i^g\right)^2}$$

44.2 Linear Discrimination Between 2 Groups

The procedure is based on the linear discriminant function of Fisher and uses the total covariance matrix for calculating coefficients of this function. Classification of cases is done using the values of this function,

and not distances as such. The criterion applied for selecting the next variable is the D^2 of Mahalanobis (Mahalanobis distance between two groups). After each step, the program provides the linear discriminant function, the classification table and the percentage of correctly classified cases for both the basic and test samples.

a) ***Linear discriminant function.*** Let us denote the function calculated in step q as

$$f_q(x) = \sum_{i \in I_q} b_{qi}\, x_i \; + a_q$$

The coefficients b_{qi} of this function for the variables i included in step q correspond to the elements of the unique eigenvector of the matrix

$$(y_q^1 - y_q^2)'\; \mathcal{T}_q^{-1}$$

and the constant term is calculated as follows:

$$a_q = -\frac{1}{2}(y_q^1 - y_q^2)'\; \mathcal{T}_q^{-1}\; (y_q^1 + y_q^2)$$

where $\mathcal{T}_q$ is the matrix of total covariance (calculated for the cases from both groups) for the variables included in step q, with the elements

$$t_{ij} = \frac{\sum\limits_k w_k\,(x_{ki} - \overline{x}_i)(x_{kj} - \overline{x}_j)}{W^1 + W^2}$$

b) ***Classification table for basic sample.***

A case is assigned:

to the group 1 if $f_q(x) > 0$,
to the group 2 if $f_q(x) < 0$.

A case is not assigned if $f_q(x) = 0$.

PERCENTAGE OF CORRECTLY CLASSIFIED CASES is calculated as the ratio between the number of cases on diagonal and the total number of cases in the classification table.

c) ***Classification table for test sample.***

Constructed in the same way as for the basic sample (see 2.b above).

d) ***Criterion for selecting the next variable.*** The Mahalanobis distance between the two groups is used for this purpose. The variable selected in step q is the one which maximizes the value of D_q^2.

$$D_q^2 = (y_q^1 - y_q^2)'\; \mathcal{T}_q^{-1}\; (y_q^1 - y_q^2)$$

e) ***Allocation and value of the linear discriminant function for the cases.*** These are calculated and printed for the last step, or when the step precedes a decrease of the percentage of correctly classified cases. The function value is calculated according to the formula described under point 2.a above; the variables used in the calculation are those retained in the step. The assignment of cases to the groups is done as described under point 2.b above.

The same formula and assignment rules are used for the basic sample, the group means, the test sample and the anonymous sample.

44.3 Linear Discrimination Between More Than 2 Groups

The procedure for discrimination of 3 or more groups uses not only the total covariance matrix but also the between groups covariance matrix. The criterion for selecting the next variable used here is the trace of a product of these two matrices (generalization of Mahalanobis distance for two groups). After selecting the new variable to be entered, discriminant factor analysis is performed and the program provides the overall discriminant power and the discriminant power of the first three factors. Cases are classified according to their distances from the centres of groups. In each step, the program calculates and prints the classification table and the percentage of correctly classified cases for both the basic and test samples.

a) ***Classification table for basic sample.*** The distance of a case x from the centre of the group g in the step q is defined as the linear function

$$v_{y_q^g}(x) = (y_q^g)' \; \mathcal{T}_q^{-1} \; (y_q^g - 2x)$$

where $\mathcal{T}_q$, as described under 2.a above, is the matrix of total covariance (calculated for the cases from all groups) for the variables included in step q, with the elements

$$t_{ij} = \frac{\sum_k w_k \, (x_{ki} - \overline{x}_i)(x_{kj} - \overline{x}_j)}{W}$$

A case is assigned to the group for which $v_{y_q^g}(x)$ has the smallest value (the smallest distance).

PERCENTAGE OF CORRECTLY CLASSIFIED CASES is calculated as the ratio between the number of cases on diagonal and the total number of cases in the classification table.

b) ***Classification table for test sample.***

Constructed in the same way as for the basic sample (see 3.a above).

c) ***Criterion for selecting the next variable.*** The variable selected in the step q is the one which maximizes the value of the trace of the matrix $\mathcal{T}_q^{-1} \; \mathcal{B}_q$, where $\mathcal{T}_q$ is the total covariance matrix used in step q (see 3.a above), and $\mathcal{B}_q$ is the matrix of covariances between groups, with the elements

$$b_{ij} = \frac{\sum_g W^g \, (y_i^g - \overline{x}_i)(y_j^g - \overline{x}_j)}{W}$$

The following part of analysis (points 3.d - 3.h below) is performed in one of the three following circumstances:

- when the step precedes a decrease of the percentage of correctly classified cases,
- when the percentage of correctly classified cases is equal to 100,
- when the step is the last one.

d) ***Allocation and distances of cases in the basic sample.*** The distances from each group are calculated as described under point 3.a above; the variables used in the calculation are those retained in the step. The assignment of cases to the groups is done as described under point 3.a above.

e) ***Discriminant factor analysis.*** The matrix $\mathcal{T}_q^{-1} \; \mathcal{B}_q$ described under 3.c above is analysed. The first two eigenvectors corresponding to the two highest eigenvalues of this matrix are the two discriminant factorial axes. The discriminant power of the factors is measured by the corresponding eigenvalues. Since the program provides the discriminant power for the first three factors, the sum of eigenvalues allows to estimate the level of remaining eigenvalues, i.e. those which are not printed.

f) ***Values of discriminant factors for all cases and group means.***

For a CASE, the value of discriminant factor is calculated as the scalar product of the case vector containing variables retained in the step by the eigenvector corresponding to the factor. Note that these values are not printed, but they are used in a graphical representation of cases in the space of the first two factors.

For a GROUP MEAN, the value of discriminant factor is calculated in the same way replacing the case vector by the group mean vector.

g) ***Allocation and distances of cases in the test sample.*** The distances from each group are calculated in the same way, and assignment of cases to the groups is done following the same rules as for the basic sample (see 3.d above).

h) ***Allocation and distances of cases in the anonymous sample.*** The distances from each group are calculated the same way and assignment of cases to the groups is done following the same rules as for the basic sample (see 3.d above).

44.4 References

Romeder, J.M., *Méthodes et programmes d'analyse discriminante*, Dunod, Paris, 1973.

Chapter 45

Distribution and Lorenz Functions

Notation

p_i = value of i^{th} break point
i = subscript for break point
s = number of subintervals
N = total number of cases.

45.1 Formula for Break Points

The number of break points is one less than the number of requested subintervals, e.g. medians imply two subintervals and one break point.

$$p_i = \mathcal{V}(\alpha) + \beta[\mathcal{V}(\alpha+1) - \mathcal{V}(\alpha)]$$

where $\mathcal{V}$ is an ordered data vector, e.g. $\mathcal{V}(3)$ is the third item in the vector,

$$\alpha = \text{entier}\left[\frac{i(N+1)}{s}\right]$$

$$\beta = \frac{i(N+1)}{s} - \alpha$$

and entier(x) is the greatest integer not exceeding x.

45.2 Distribution Function Break Points

There are four possible situations:

- If a break point falls exactly on a value and the value is not tied with any other value, then the value itself is the break point.
- If a break point falls between two values and the two values are not the same, then the break point is determined using ordinary linear interpolation.
- If a break point falls exactly on a value and the value is tied with one or more other values, then the procedure involves computing new midpoints. Let k be the value, m be the frequency with which it occurs and d be the minimum distance between items in the vector $\mathcal{V}$. The interval $k \pm \min(d,1)/2$ is divided into m parts and midpoints are computed for these new intervals. The break point is then the appropriate midpoint.
- If a break point falls between two values which are identical, the procedure involves both the calculation of new midpoints and ordinary linear interpolation. Let k be the value, m be the frequency with which

it occurs and d be the minimum distance between items in the vector $\mathcal{V}$. The interval $k \pm \min(d, 1)/2$ is divided into m parts and midpoints are computed for these new intervals. Then linear interpolation is performed between the two appropriate new midpoints.

45.3 Lorenz Function Break Points

To determine Lorenz function break points, the ordered data vector is cumulated, and at each step the cumulated total is divided by the grand total. Then the break points are found the same way as described above.

45.4 Lorenz Curve

The Lorenz function plotted against the proportion of the ordered population gives a Lorenz curve, which is always contained in the lower triangle of the unit square. The QUANTILE program uses ten subintervals for the Lorenz curve.

Note that Lorenz function values are called "Fraction of wealth" on the printout.

45.5 The Gini Coefficient

The Gini coefficient represents twice the area between the Lorenz function and the diagonal plotted in the unit square. It takes on values between 0 and 1. Zero (0) indicates "perfect equality" - all data values are equal. One (1) indicates "perfect inequality" - there is one non-zero data value.

The program uses an approximation:

$$\text{Gini coefficient} = 1 - \frac{1}{s} - \frac{2}{s}\sum_{i=1}^{s-1} l_i$$

where l_i is the i^{th} Lorenz function break point.

This approximation becomes more accurate as the number of break points is increased; it is recommended that at least ten be used.

45.6 Kolmogorov-Smirnov D Statistic

The Kolmogorov-Smirnov test is concerned with the agreement between two cumulative distributions. If two sample cumulative distributions are too far apart at any point, it suggests that the samples come from different populations. The test focuses on the largest difference between the two distributions.

Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be the ordered data vectors for the first and the second variable respectively, and $\mathcal{X}$ the vector of codes which appear in either distribution. The program creates the two cumulative step functions $F_1(x)$ and $F_2(x)$ respectively. Then it looks for maximum absolute difference between the distributions,

$$D = \max(|F_1(x) - F_2(x)|)$$

and prints:

x : the value where the first maximum absolute difference occurs
f_1 : the value of F_1 associated with the x
f_2 : the value of F_2 associated with the x.

If the N's for $\mathcal{V}_1$ and $\mathcal{V}_2$ are equal and less than 40, the program prints K statistic equal to the difference in frequencies associated with the maximum difference. A table of critical values of K statistic, denoted K_D, can be consulted to determine the significance of the observed difference.

If the N's for $\mathcal{V}_1$ and $\mathcal{V}_2$ are unequal or larger than 40, the program prints the following statistics:

$$\text{Unadjusted deviation} = D = |f_1 - f_2|$$

$$\text{Adjusted deviation} = D\sqrt{\frac{N_1\,N_2}{N_1 + N_2}}$$

where N_1 and N_2 are equal to the number of cases in $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively.

$$\text{Chi-squared approximation} = 4D^2\,\frac{N_1\,N_2}{N_1 + N_2}$$

Note: The significance of the maximum directional deviation can be found by referring this chi-square value to a chi-square distribution with two degrees of freedom.

45.7 Note on Weights

For distribution function break points, Lorenz function break points, and the Gini coefficients, data may be weighted by an integer. If a weight is specified, each case is implicitly counted as "w" cases, where "w" is the weight value for the case. The Kolmogorov-Smirnov test is always performed on unweighted data.

Chapter 46

Factor Analyses

Notation

x	=	values of variables
i	=	subscript for case
j, j'	=	subscripts for variables
α	=	subscript for factor
m	=	number of factors determined/desired
$I1$	=	number of principal cases
$J1$	=	number of principal variables
w	=	value of the weight
W	=	total sum of weights for principal cases.

46.1 Univariate Statistics

These univariate statistics are calculated for all variables used in the analysis, i.e. principal and supplementary variables, if any. Note that variables are renumbered from 1 (column RNK). Only principal cases enter into calculations.

a) ***Mean.***

$$\overline{x}_j = \frac{\sum_{i=1}^{I1} w_i \, x_{ij}}{W}$$

b) ***Variance (estimated).***

$$\widehat{s_j}^2 = \left(\frac{N}{N-1}\right)\left[\frac{W\sum_{i=1}^{I1} w_i \, x_{ij}^2 - \left(\sum_{i=1}^{I1} w_i \, x_{ij}\right)^2}{W^2}\right]$$

c) ***Standard deviation (estimated).***

$$\widehat{s_j} = \sqrt{\widehat{s_j}^2}$$

d) ***Coefficient of variability (C. Var.).***

$$C_j = \frac{\widehat{s_j}}{\overline{x}_j}$$

e) ***Total (sum for x_j).***

$$Total_j = \sum_{i=1}^{I1} w_i \, x_{ij}$$

f) ***Skewness.***

$$g1_j = \frac{m3_j}{\widehat{s}_j^2 \sqrt{\widehat{s}_j^2}} \quad \text{where} \quad m3_j = \frac{\sum_{i=1}^{I1} w_i \, (x_{ij} - \overline{x}_j)^3}{W}$$

g) ***Kurtosis.***

$$g2_j = \frac{m4_j}{(\widehat{s}_j^2)^2} - 3 \quad \text{where} \quad m4_j = \frac{\sum_{i=1}^{I1} w_i \, (x_{ij} - \overline{x}_j)^4}{W}$$

h) ***Weighted N.*** Number of principal cases if the weight is not specified, or weighted number of principal cases (sum of weights).

46.2 Input Data

The data are printed for both principal and supplementary cases.

The first column of the table contains the values of the case ID variable (up to 4 digits). The second column (Coef) contains the value of the weight assigned to each case (w_i). The third column (PI) is equal to the weighted sum of principal variables' values, for each case (weighted row totals).

$$P_{i\cdot} = \sum_{j=1}^{J1} w_i \, x_{ij}$$

The first line contains the first four characters of each variable name. The second line (PJ) is equal to the weighted sum of principal cases' values, for each variable (weighted column totals).

$$P_{\cdot j} = \sum_{i=1}^{I1} w_i \, x_{ij}$$

Note that the value of the "Coef" at the beginning of this line is equal to the weighted number of principal cases, and the value of "PI" is equal to the overall Total (P) of the principal variables for the principal cases.

$$P = \sum_{i=1}^{I1} P_{i\cdot} = \sum_{j=1}^{J1} P_{\cdot j} = \sum_{i=1}^{I1} \sum_{j=1}^{J1} w_i \, x_{ij}$$

The rest of the input data table contains the values (with one decimal point) of principal and supplementary variables.

46.3 Core Matrices (Matrices of Relations)

For each type of analysis, a core matrix is calculated and printed. This is a matrix of relationships between variables. Note that for the printout, the values in the matrix are multiplied by a factor the value of which is printed next to the matrix title. This factor is set to zero when some values in the matrix exceed 5 characters (it may be the case of scalar products or covariances matrices).

For the ANALYSIS OF CORRESPONDENCES, the elements $C_{jj'}$ of the core matrix are calculated as follows:

$$C_{jj'} = \frac{1}{\sqrt{P_{\cdot j}} \sqrt{P_{\cdot j'}}} \sum_{i=1}^{I1} \frac{(w_i \, x_{ij}) \, (w_i \, x_{ij'})}{P_{i\cdot}}$$

For the ANALYSIS OF SCALAR PRODUCTS, the elements $SP_{jj'}$ of the core matrix are calculated as follows:

$$SP_{jj'} = \sum_{i=1}^{I1} w_i \, x_{ij} \, x_{ij'}$$

For the ANALYSIS OF NORMED SCALAR PRODUCTS, the elements $NSP_{jj'}$ of the core matrix are calculated as follows:

$$NSP_{jj'} = \frac{\sum_{i=1}^{I1} w_i \, x_{ij} \, x_{ij'}}{\sqrt{\left(\sum_{i=1}^{I1} w_i \, x_{ij}^2\right)\left(\sum_{i=1}^{I1} w_i \, x_{ij'}^2\right)}}$$

For the ANALYSIS OF COVARIANCES, the elements $COV_{jj'}$ of the core matrix are calculated as follows:

$$COV_{jj'} = \frac{\sum_{i=1}^{I1} w_i \, (x_{ij} - \overline{x}_j) \, (x_{ij'} - \overline{x}_{j'})}{W}$$

For the ANALYSIS OF CORRELATIONS, the elements $COR_{jj'}$ of the core matrix are calculated as follows:

$$COR_{jj'} = \frac{\sum_{i=1}^{I1} w_i \, (x_{ij} - \overline{x}_j) \, (x_{ij'} - \overline{x}_{j'})}{\sqrt{\sum_{i=1}^{I1} w_i \, (x_{ij} - \overline{x}_j)^2 \; \sum_{i=1}^{I1} w_i \, (x_{ij'} - \overline{x}_{j'})^2}}$$

46.4 Trace

Trace of the core matrix is calculated as a sum of its diagonal elements. Trace is also equal to the total of eigenvalues (total inertia). Note that for the analysis of correlations and the analysis of normed scalar products the total inertia is equal to the number of principal variables.

$$Trace = \sum_{\alpha=1}^{J1} \lambda_\alpha$$

46.5 Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors are printed for the factors retained. They have the same meaning for each type of analysis but they are of little interest for the user.

For analysis of correspondences, the program prints here one eigenvalue and eigenvector more than the number of factors determined/desired. The factor for the trivial eigenvalue (being always equal to 1) is printed as the first one and is neglected later on. The remaining factors are renumbered (starting from 1) in the tables of principal/supplementary variables/cases.

46.6 Table of Eigenvalues

The table contains all the eigenvalues, denoted here by λ_α, calculated by the program. Note that in analysis of correspondences, the first, trivial eigenvalue (being always 1) is printed only over the table and its value is subtracted from the Trace in calculating the percent in the point 6.d below.

a) ***NO.*** Eigenvalue sequential number, α, in ascending order.

b) ***ITER.*** Number of iterations used in computing corresponding eigenvectors. Value zero means that the corresponding eigenvector was obtained at the same time that the previous one (from the bottom).

c) ***Eigenvalue.*** This column gives a sequence of eigenvalues, lambdas, each corresponding to the factor α.

d) ***Percent.*** Contribution of the factor to the total inertia (in terms of percentages).

$$\tau_\alpha = \frac{\lambda_\alpha}{\text{Trace}} \times 100$$

e) ***Cumul (cumulative percent).*** Contribution of the factors 1 through α to the total inertia (in terms of percentages).

$$\text{Cumul}_\alpha = \tau_1 + \tau_2 + \cdots + \tau_\alpha$$

f) ***Histogram of eigenvalues.*** Each eigenvalue is represented by a line of asterisks the number of which is proportional to the eigenvalue. The first eigenvalue in the histogram is always represented by 60 asterisks. The histogram permits a visual analysis of the relative diminution of eigenvalues for subsequent factors.

46.7 Table of Principal Variables' Factors

The table contains the ordinates of the principal variables in the factorial space, their squared cosines with each factor and their contributions to each factor. In addition, it contains the quality of these variables, their weights and their inertia.

a) ***JPR.*** Variable number for the principal variables.

b) ***QLT.*** Quality of representation of the variable in the space of m factors is measured, for ALL TYPES OF ANALYSIS, by the sum of the squared cosines (see 7.f below). Values closer to 1 indicate higher level of representation of the variable by the factors.

$$QLT_j = \sum_{\alpha=1}^{m} COS2_{\alpha\, j}$$

c) ***WEIG.*** Weight value of the variable. For ALL TYPES OF ANALYSIS, it is calculated as a ratio between the total of the variable and the overall Total (see section 2 above), multiplied by 1000.

$$f_{\cdot j} = \frac{P_{\cdot j}}{P} \times 1000$$

Note that the weight (WEIG) printed in the last line of the table is equal to:

- the overall Total for the correspondence analysis,
- the weighted number of cases for other types of analysis.

d) ***INR.*** Inertia corresponding to the variable. It indicates the part of the total inertia related to the variable in the space of factors.

For the ANALYSIS OF CORRESPONDENCES, it is calculated as a ratio between the inertia of the variable and the total inertia, multiplied by 1000. Note that the inertia of the variable depends on the variable weight and that the Trace value used here does not include the trivial eigenvalue.

$$INR_j = \frac{f_{\cdot j} \sum_{\alpha=1}^{J1-1} F_{\alpha\, j}^2}{Trace} \times 1000$$

where $F_{\alpha\, j}$ is the ordinate of the variable j corresponding to the factor α (see 7.e below).

For the ANALYSIS OF SCALAR PRODUCTS and the ANALYSIS OF COVARIANCES, the inertia of the variable does not depend on the variable weight.

$$INR_j = \frac{\sum_{\alpha=1}^{J1} F_{\alpha\, j}^2}{Trace} \times 1000$$

For the ANALYSIS OF NORMED SCALAR PRODUCTS and the ANALYSIS OF CORRELATIONS, the inertia of the variable depends only on the number of principal variables.

$$INR_j = \frac{1}{J1} \times 1000$$

Note that the inertia (INR) printed in the last line of the table is equal to 1000.

The three following columns are repeated for each factor.

e) $\alpha\#F$**.** The ordinate of the variable in the factor space, denoted here by $F_{\alpha\, j}$.

f) ***COS2.*** Squared cosine of the angle between the variable and the factor. It is a measure of "distance" between the variable and the factor. Values closer to 1 indicate shorter distances from the factor.

For the ANALYSIS OF CORRESPONDENCES, it is calculated as follows:

$$COS2_{\alpha\, j} = \frac{F_{\alpha\, j}^2}{\sum_{\alpha=1}^{J1-1} F_{\alpha\, j}^2} \times 1000$$

For the ANALYSIS OF SCALAR PRODUCTS and the ANALYSIS OF COVARIANCES,

$$COS2_{\alpha\, j} = \frac{F_{\alpha\, j}^2}{\sum_{\alpha=1}^{J1} F_{\alpha\, j}^2} \times 1000$$

For the ANALYSIS OF NORMED SCALAR PRODUCTS and the ANALYSIS OF CORRELATIONS,

$$COS2_{\alpha\, j} = F_{\alpha\, j}^2 \times 1000$$

g) ***CPF.*** Contribution of the variable to the factor.

For the ANALYSIS OF CORRESPONDENCES,

$$CPF_{\alpha\, j} = \frac{f_{\cdot j}\, F_{\alpha\, j}^2}{\lambda_\alpha} \times 1000$$

For ALL THE OTHER TYPES OF ANALYSIS,

$$CPF_{\alpha\, j} = \frac{F_{\alpha\, j}^2}{\lambda_\alpha} \times 1000$$

Note that the contribution (CPF) printed in the last line of the table is equal to 1000.

46.8 Table of Supplementary Variables' Factors

The table contains the same information as the one described under point 7. above, but for the supplementary variables.

a) ***JSUP.*** Variable number for the supplementary variables.

b) ***QLT.*** Quality of representation of the variable in the space of m factors (see 7.b above).

c) ***WEIG.*** Weight value of the variable (see 7.c above).

d) ***INR.*** Inertia corresponding to the variable. Note that the supplementary variables do not contribute to the total inertia. Thus, the inertia here indicates whether the variable could play any role in the analysis if it would be used as a principal one. It is calculated in the same way as for the principal variables in respective analyses (see 7.d above).

The inertia (INR) printed in the last line of the table is equal to the total INR over all the supplementary variables.

The three following columns are repeated for each factor.

e) $\alpha\#F$**.** The ordinate of the variable in the factor space, denoted here by $F_{\alpha\, j}$.

f) ***COS2.*** Squared cosine of the angle between the variable and the factor. It is calculated in the same way as for the principal variables in respective analyses (see 7.f above).

g) ***CPF.*** Contribution of the variable to the factor. Note that the supplementary variables do not participate in the construction of the factor space. Thus, the contribution only indicates whether the variable could play any role in the analysis if it would be used as a principal one. CPF is calculated in the same way as for the principal variables in respective analyses (see 7.g above).

The contribution (CPF) printed in the last line of the table is equal to the total CPF over all the supplementary variables.

46.9 Table of Principal Cases' Factors

The table contains the ordinates of the principal cases in the factorial space, their squared cosines with each factor and their contributions to each factor. In addition, it contains the quality of representation of these cases, their weights and their inertia.

a) ***IPR.*** Case ID value for the principal cases.

b) ***QLT.*** Quality of representation of the case in the space of m factors is measured, for ALL TYPES OF ANALYSIS, by the sum of the squared cosines (see 9.f below). Values closer to 1 indicate higher level of representation of the case by the factors.

$$QLT_i = \sum_{\alpha=1}^{m} COS2_{\alpha\, i}$$

c) ***WEIG.*** Weight value of the case.

For the ANALYSIS OF CORRESPONDENCES, it is calculated as a ratio between the (weighted) sum of principal variables for this case and the overall Total (see section 2 above), multiplied by 1000.

$$f_{i\cdot} = \frac{P_{i\cdot}}{P} \times 1000$$

Note that the weight (WEIG) printed in the last line of the table is equal to the overall Total.

For ALL OTHER TYPES OF ANALYSIS,

$$f_{i\cdot} = \frac{w_i}{P} \times 1000$$

Note that the weight (WEIG) printed in the last line of the table is equal to the weighted number of cases.

d) ***INR.*** Inertia corresponding to the case. It indicates the part of the total inertia related to the case in the space of factors.

For the ANALYSIS OF CORRESPONDENCES, it is calculated as a ratio between the inertia of the case and the total inertia, multiplied by 1000. Note that the inertia of the case depends on the case weight and that the Trace value used here does not include the trivial eigenvalue.

$$INR_i = \frac{f_{i\cdot} \sum_{\alpha=1}^{J1-1} F_{\alpha\, i}^2}{Trace} \times 1000$$

For ALL OTHER TYPES OF ANALYSIS,

$$INR_i = \left(\frac{w_i}{W \times Trace} \sum_{j=1}^{J1} z_{ij}^2 \right) \times 1000$$

where

$$z_{ij} = \begin{cases} x_{ij} & \text{for analysis of scalar products} \\ \dfrac{x_{ij}}{\sqrt{\left(\sum_{i=1}^{I1} w_i\, x_{ij}^2\right) / W}} & \text{for analysis of normed scalar products} \\ x_{ij} - \overline{x}_j & \text{for analysis of covariances} \\ \dfrac{x_{ij} - \overline{x}_j}{s_j} & \text{for analysis of correlations} \end{cases}$$

and s_j is the sample standard deviation of the variable j.

Note that the inertia (INR) printed in the last line of the table is equal to 1000.

The three following columns are repeated for each factor.

e) $\alpha\#F$**.** The ordinate of the case in the factor space, denoted here by $F_{\alpha\, i}$.

f) ***COS2.*** Squared cosine of the angle between the case and the factor. It is a measure of "distance" between the case and the factor. Values closer to 1 indicate shorter distances from the factor.

For the ANALYSIS OF CORRESPONDENCES, it is calculated as follows:

$$COS2_{\alpha\, i} = \frac{F_{\alpha\, i}^2}{\sum_{\alpha=1}^{J1-1} F_{\alpha\, i}^2} \times 1000$$

For ALL OTHER TYPES OF ANALYSIS,

$$COS2_{\alpha\, i} = \frac{F_{\alpha\, i}^2}{\sum_{\alpha=1}^{J1} F_{\alpha\, i}^2} \times 1000$$

g) ***CPF.*** Contribution of the case to the factor.

For the ANALYSIS OF CORRESPONDENCES,

$$CPF_{\alpha\, i} = \frac{f_{i\cdot}\, F_{\alpha\, i}^2}{\lambda_\alpha} \times 1000$$

For ALL OTHER TYPES OF ANALYSIS,

$$CPF_{\alpha\, i} = \frac{w_i\, F_{\alpha\, i}^2}{W\, \lambda_\alpha} \times 1000$$

Note that the contribution (CPF) printed in the last line of the table is equal to 1000.

46.10 Table of Supplementary Cases' Factors

The table contains the same information as the one described under the point 9. above, but for the supplementary cases.

a) ***ISUP.*** Case ID value for the supplementary cases.

b) ***QLT.*** Quality of representation of the case in the space of m factors (see 9.b above).

c) ***WEIG.*** Weight value of the case (see 9.c above).

d) ***INR.*** Inertia corresponding to the case. Note that the supplementary cases do not contribute to the total inertia. Thus, the inertia here indicates whether the case could play any role in the analysis if it would be used as a principal one. It is calculated the same way as for the principal cases in respective analyses (see 9.d above).

The inertia (INR) printed in the last line of the table is equal to the total INR over all the supplementary cases.

The three following columns are repeated for each factor.

e) $\alpha\#F$***.*** The ordinate of the case in the factor space, denoted here by $F_{\alpha\, i}$.

f) ***COS2.*** Squared cosine of the angle between the case and the factor. It is calculated the same way as for the principal cases in respective analyses (see 9.f above).

g) ***CPF.*** Contribution of the case to the factor. Note that the supplementary cases do not participate in the construction of the factor space. Thus, the contribution only indicates whether the case could play any role in the analysis if it would be used as a principal one. CPF is calculated the same way as for the principal cases in respective analyses (see 9.g above).

The contribution (CPF) printed in the last line of the table is equal to the total CPF over all the supplementary cases.

46.11 Rotated Factors

Applied only for correlation analysis. The "variable" factors can be rotated once the factor analysis is terminated. The Varimax procedure used here is the same as the one used in CONFIG program. Note that the "variable" factors for principal variables may be treated as a configuration of $J1$ objects in α dimensional space.

46.12 References

Benzécri, J.-P. and F., *Pratique de l'analyse de données*, tome 1: *Analyse des correspondances*, exposé élémentaire, Dunod, Paris, 1984.

Iagolnitzer, E.R., Présentation des programmes MLIFxx d'analyses factorielles en composantes principales, *Informatique et sciences humaines*, 26, 1975.

Chapter 47

Linear Regression

Notation

y	=	value of the dependent variable
x	=	value of an independent (explanatory) variable
i, j, l, m	=	subscripts for variables
p	=	number of predictors
k	=	subscript for case
N	=	total number of cases
w	=	value of the weight multiplied by $\frac{N}{W}$
W	=	total sum of weights.

47.1 Univariate Statistics

These weighted statistics are calculated for all variables used in the analysis, i.e. dummy variables, independent variables and the dependent variable.

a) ***Average.***

$$\overline{x}_i = \frac{\sum_k w_k \, x_{ik}}{N}$$

b) ***Standard deviation (estimated).***

$$\widehat{s}_i = \sqrt{\frac{N \sum_k (w_k \, x_{ik})^2 \; - \; \left(\sum_k w_k \, x_{ik} \right)^2}{N(N-1)}}$$

c) ***Coefficient of variation (C.var.).***

$$C_i = \frac{100 \, \widehat{s}_i}{\overline{x}_i}$$

47.2 Matrix of Total Sums of Squares and Cross-products

It is calculated for all variables used in the analysis as follows:

$$\text{t.s.s.c.p.}_{\,ij} = \sum_k w_k \, x_{ik} \, x_{jk}$$

47.3 Matrix of Residual Sums of Squares and Cross-products

This matrix, sometimes called a matrix of squares and cross-products of deviation scores, is calculated for all variables used in the analysis as follows:

$$\text{r.s.s.c.p.}_{ij} = \sum_k w_k\, x_{ik}\, x_{jk} \;-\; \frac{\left(\sum_k w_k\, x_{ik}\right)\left(\sum_k w_k\, x_{jk}\right)}{N}$$

47.4 Total Correlation Matrix

The elements of this matrix are calculated directly from the matrix of residual sums of squares and cross products. Note that if this formula is written out in detail, and the numerator and denominator are both multiplied by N, it is a conventional formula for Pearson's r.

$$r_{ij} = \frac{\text{r.s.s.c.p.}_{ij}}{\sqrt{\text{r.s.s.c.p.}_{ii}}\,\sqrt{\text{r.s.s.c.p.}_{jj}}}$$

47.5 Partial Correlation Matrix

The ij^{th} element of this matrix is the partial correlation coefficient between variable i and variable j, holding constant specified variables. Partial correlations describe the degree of correlation that would exist between two variables provided that variation in one or more other variables is controlled. They also describe the correlation between independent (explanatory) variables which would be selected in a stepwise regression.

a) ***Correlation between*** x_i ***and*** x_j ***holding constant*** x_l (first-order partial correlation coefficients).

$$r_{ij\cdot\,l} = \frac{r_{ij} \;-\; r_{il}\, r_{jl}}{\sqrt{1 - r_{il}^2}\,\sqrt{1 - r_{jl}^2}}$$

where r_{ij}, r_{il}, r_{jl} are zero-order coefficients (Pearson's r coefficients).

b) ***Correlation between*** x_i ***and*** x_j ***holding constant*** x_l ***and*** x_m (second-order partial correlation coefficients).

$$r_{ij\cdot\,lm} = \frac{r_{ij\cdot\,l} \;-\; r_{im\cdot\,l}\, r_{jm\cdot\,l}}{\sqrt{1 - r_{im\cdot\,l}^2}\,\sqrt{1 - r_{jm\cdot\,l}^2}}$$

where $r_{ij\cdot\,l}$, $r_{im\cdot\,l}$, $r_{jm\cdot\,l}$ are first-order coefficients.

Note: The program computes the partial correlations by working up step by step from zero-order coefficients to first order, to second order, etc.

47.6 Inverse Matrix

For a standard regression, this is the inverse of the correlation matrix of the independent (explanatory) variables and the dependent variable. For a stepwise regression, this is the inverse of the correlation matrix of the independent variables in the final equation. The program uses the Gaussian elimination method for inverting.

47.7 Analysis Summary Statistics

a) ***Standard error of estimate.*** This is the standard deviation of the residuals.

$$\text{Standard error of estimate} = \sqrt{\frac{\sum_k (y_k - \widehat{y}_k)^2}{\text{df}}}$$

where

$\widehat{y}_k$ = the predicted value of the dependent variable for the k^{th} case
df = residual degrees of freedom (see 7.f below).

b) ***F-ratio for the regression.*** This is the F statistic for determining the statistical significance of the model under consideration. The degrees of freedom are p and $N - p - 1$.

$$F = \frac{R^2 \,\text{df}}{p\,(1 - R^2)}$$

where R^2 is the fraction of explained variance (see 7.d below).

c) ***Multiple correlation coefficient.*** This is the correlation between the dependent variable and the predicted score. It indicates the strength of relationship between the criterion and the linear function of the predictors, and is similar to a simple Pearson correlation coefficient except that it is always positive.

$$R = \sqrt{R^2}$$

R is not printed if the constant term is constrained to be zero.

d) ***Fraction of explained variance.*** R^2 can be interpreted as the proportion of variation in the dependent variable explained by the predictors. Sometimes called the coefficient of determination, it is a measure of the overall effectiveness of the linear regression. The larger it is, the better the fitted equation explains the variation in the data.

$$R^2 = 1 - \frac{\sum_k (y_k - \widehat{y}_k)^2}{\sum_k (y_k - \overline{y})^2}$$

where

$\widehat{y}_k$ = the predicted value of the dependent variable for the k^{th} case
$\overline{y}$ = the mean of the dependent variable.

Like R, R^2 is not printed if the constant term is constrained to be zero.

e) ***Determinant of the correlation matrix.*** This is the determinant of the correlation matrix of the predictors. It represents as a single number the generalized variance in a set of variables, and varies from 0 to 1. Determinants near zero indicate that some or all explanatory variables are highly correlated. A zero determinant indicates a singular matrix, which means that at least one of the predictors is a linear function of one or more others.

f) ***Residual degrees of freedom.***

If the constant is not constrained to be zero,

$$\text{df} = N - p - 1$$

If the constant is constrained to be zero,

$$\text{df} = N - p$$

g) ***Constant term.***

$$A = \overline{y} - \sum_i B_i \, \overline{x}_i$$

where

$\overline{y}$ = the average of the dependent variable (see 1.a above)
$\overline{x}_i$ = the average of the predictor variable i (see 1.a above)
B_i = the B coefficient for the predictor variable i (see 8.a below).

47.8 Analysis Statistics for Predictors

a) ***B.*** These are unstandardized partial regression coefficients which are appropriate (rather than the betas) to be used in an equation to predict raw scores. They are sensitive to the scale of measurement of the predictor variable and to the variance of the predictor variable.

$$B_i = \beta_i \, \frac{\widehat{s}_y}{\widehat{s}_i}$$

where

β_i = the beta weight for predictor i (see 8.c below)
$\widehat{s}_y$ = the standard deviation of the dependent variable (see 1.b above)
$\widehat{s}_i$ = the standard deviation of the predictor variable i (see 1.b above).

b) ***Sigma B.*** This is the standard error of B, a measure of the reliability of the coefficient.

$$\text{Sigma } B_i = (\text{standard error of estimate}) \sqrt{\frac{c_{ii}}{\text{r.s.s.c.p.}_{\,ii}}}$$

where c_{ii} is the i^{th} diagonal element of the inverse of the correlation matrix of predictors in the regression equation (see section 6 above).

c) ***Beta.*** These regression coefficients are also called "standardized partial regression coefficients" or "standardized B coefficients". They are independent from a scale of measurement. The magnitudes of the squares of the betas indicate the relative contributions of the variables to the prediction.

$$\beta_i = \mathcal{R}_{11}^{-1} \, \mathcal{R}_{yi}$$

where

$\mathcal{R}_{11}$ = correlation matrix of predictors in the equation
$\mathcal{R}_{yi}$ = column vector of correlations of the dependent variable and predictors indicated by the predictor i.

d) ***Sigma Beta.*** This is the standard error of the beta coefficient, a measure of the reliability of the coefficient.

$$\text{Sigma } \beta_i = \text{sigma } B_i \, \frac{\widehat{s}_i}{\widehat{s}_y}$$

e) ***Partial r squared.*** These are partial correlations, squared, between predictor i and the dependent variable, y, with the influence of the other variables in the regression equation eliminated. The partial correlation coefficient squared is a measure of the extent to which that part of the variation in the dependent variable which is not explained by the other predictors is explained by predictor i.

$$r^2_{yi \cdot \, jl\ldots} = \frac{R^2_{y \cdot \, ijl\ldots} - R^2_{y \cdot \, jl\ldots}}{1 - R^2_{y \cdot \, jl\ldots}}$$

where

$$R^2_{y\cdot\, ijl\ldots} = \text{multiple } R \text{ squared with predictor } i$$
$$R^2_{y\cdot\, jl\ldots} = \text{multiple } R \text{ squared without predictor } i.$$

f) ***Marginal r squared.*** This is the increase in variance explained by adding predictor i to the other predictors in the regression equation.

$$\text{marginal}\, r_i^2 = R^2_{y\cdot\, ijl\ldots} - R^2_{y\cdot\, jl\ldots}$$

g) ***The t-ratio.*** It can be used to test the hypothesis that β, or B, is equal to zero; that is, that predictor i has no linear influence on the dependent variable. Its significance can be determined from the table of t, with $N - p - 1$ degrees of freedom.

$$t = \left| \frac{\beta_i}{\text{sigma}\, \beta_i} \right| = \left| \frac{B_i}{\text{sigma}\, B_i} \right|$$

h) ***Covariance ratio.*** The covariance ratio of x_i is the square of the multiple correlation coefficient, R^2, of x_i with the $p-1$ other independent variables in the equation. It is a measure of the intercorrelation of x_i with the other predictors.

$$\text{Covariance ratio}_{\, i} = 1 - \frac{1}{c_{ii}}$$

where c_{ii} is the i^{th} diagonal element of the inverse of the correlation matrix of predictors in the regression equation (see section 6 above).

47.9 Residuals

The residuals are the difference between the observed value of the dependent variable and the value predicted by the regression equation.

$$e_k = y_k - \widehat{y}_k$$

The test for detecting serial correlation, popularly known as the Durbin-Watson d statistic for first-order autocorrelation of residuals, is calculated as follows:

$$d = \frac{\sum_{k=2}^{N} (e_k - e_{k-1})^2}{\sum_{k=1}^{N} e_k^2}$$

47.10 Note on Stepwise Regression

Stepwise regression introduces the predictors step by step into the model, starting with the independent variable most highly correlated with y. After the first step, the algorithm selects from the remaining independent variables the one which yields the largest reduction in the residual (unexplained) variance of the dependent variable, i.e. the variable whose partial correlation with y is the highest. The program then does a partial F-test for entrance to see if the variable will take up a significant amount of variation over that removed by variables already in the regression. The user can specify a minimum F-value for the inclusion of any variable; the program evaluates whether or not the F-value obtained at a given step satisfies the minimum, and if it does, enters the variable. Similarly, the program decides at each step whether or not each previously-included variable still satisfies a minimum (also provided by the user), and if not, removes it.

$$\text{Partial F-value for variable}\, i = \frac{(R^2_{y\cdot\, Pi} - R^2_{y\cdot\, P})(\text{df})}{1 - R^2_{y\cdot\, Pi}}$$

where

$R^2_{y\cdot Pi}$ = multiple R squared for the set of predictors (P) already in the regression, with predictor i

$R^2_{y\cdot P}$ = multiple R squared for the set of predictors (P) already in the regression

df = residuals degrees of freedom.

At any step in the procedure, the results are the same as they would be for a standard regression using the particular set of variables; thus, the final step of a stepwise regression shows the same coefficients as a normal execution using the variables that "survived" the stepwise procedure.

47.11 Note on Descending Regression

Descending regression is like the stepwise regression, except that the algorithm starts with all the independent variables and then drops and adds back variables in a stepwise manner.

47.12 Note on Regression with Zero Intercept

It is possible when using the REGRESSN program to request a zero regression intercept, i.e. that the dependent variable is zero when all the independent variables are zero.

If a regression through the origin is specified, all statistics except those described in sections 1 through 4 above are based on a mean of zero. The multiple correlation coefficient and fraction of explained variance (items 7.c and 7.d) are not printed at all. Statistics which are not centered about the mean can be very different from what they would be if they were centered; thus, in a stepwise solution, variables may very well enter the equation in a different order than they would if a constant were estimated.

In the REGRESSN program a matrix with elements

$$a_{ij} = \frac{\sum_k w_k\, x_{ik}\, x_{jk}}{\sqrt{\sum_k w_k\, x_{ik}^2 \sum_k w_k\, x_{jk}^2}}$$

is analyzed rather than $\mathcal{R}$, the correlation matrix.

The B's, the unstandardized partial regression coefficients, are obtained by

$$B_i = \beta_i \sqrt{\sum_k w_k\, x_{ik}^2 \sum_k w_k\, x_{jk}^2}$$

Chapter 48

Multidimensional Scaling

Notation

x	=	element of the configuration
i, j, l, m	=	subscripts for variables
n	=	number of variables
s	=	subscript for dimension
t	=	number of dimensions.

48.1 Order of Computations

For a given number of dimensions, t, MDSCAL finds the configuration of minimum stress by using an iterative procedure. The program starts with an initial configuration (provided by the user or by the program) and keeps modifying it until it converges to the configuration having minimum stress.

48.2 Initial Configuration

If the user does not supply a starting configuration the program generates an arbitrary configuration by taking the first n points from the following list (each expression between parenthesis represents a point):

$$
\begin{array}{c}
(1, 0, 0, \ldots, 0), \\
(0, 2, 0, \ldots, 0), \\
(0, 0, 3, \ldots, 0), \\
\vdots \\
(0, 0, 0, \ldots, t), \\
(t+1, 0, 0, \ldots, 0), \\
(0, t+2, 0, \ldots, 0), \\
\vdots
\end{array}
$$

48.3 Centering and Normalization of the Configuration

At the start of each iteration the configuration is centered and normalized.

If x_{is} denotes the element in the i^{th} line and s^{th} column of the configuration, then

$$\text{Centered } x_{is} = x_{is} - \overline{x}_s$$

$$\text{Normalized } x_{is} = \frac{x_{is} - \overline{x}_s}{\text{n.f.}}$$

where

$$\overline{x}_s = \frac{\sum_i x_{is}}{n}$$

is the mean of dimension s and

$$\text{n.f.} = \sqrt{\frac{n}{\sum_i \sum_s x_{is}^2}}$$

is the normalization factor.

Note that the total sum of squares of the elements of the normalized centered configuration is equal to n, the number of variables.

48.4 History of Computation

At the conclusion of each iteration, items 4.a through 4.h below are printed. This creates a history which, in general, is of interest only when it is feared that convergence is not complete. However, at the end of history the reason for stopping is printed. If the program does not stop because a minimum has been reached, it may nonetheless be true that the solution reached is practically indistinguishable from the minimum that would be reached after a few more iterations - in particular, if the stress is very small, this is generally the case.

a) ***Stress.*** The measure of stress serves two functions. First, it is a measure of how well the derived configuration matches the input data. Second, it is used in deciding how points should be moved on the next iteration. There are two available formulas for calculating stress: SQDIST and SQDEV.

$$\text{Stress SQDIST} = \sqrt{\frac{\sum_i \sum_j (d_{ij} - \widehat{d}_{ij})^2}{\sum_i \sum_j d_{ij}^2}}$$

$$\text{Stress SQDEV} = \sqrt{\frac{\sum_i \sum_j (d_{ij} - \widehat{d}_{ij})^2}{\sum_i \sum_j (d_{ij} - \overline{d}\,)^2}}$$

where

d_{ij} = distance between variables i and j in the configuration (see 8.c below)

$\widehat{d}_{ij}$ = those numbers which minimize the stress, subject to the constraint that the d_{ij} have the same rank order as the input data (see 8.d below)

$\overline{d}$ = the mean of all the d_{ij}'s.

b) ***SRAT.*** Stress ratio. The user can stop the scaling procedure by specifying the stress ratio to be reached. For the first iteration (numbered 0) its value is set to 0.800 .

$$\text{SRAT} = \frac{\text{Stress}_{\text{present}}}{\text{Stress}_{\text{previous}}}$$

c) ***SRATAV.*** Average stress ratio. For the first iteration its value is equal to 0.800 .

$$\text{SRATAV}_{\text{present}} = (\text{SRAT}_{\text{present}})^{0.33334} \times (\text{SRATAV}_{\text{previous}})^{0.66666}$$

d) ***CAGRGL.*** This is the cosine of the angle between the current gradient and the previous gradient.

$$\text{CAGRGL} = \cos\Theta = \frac{\sum_i \sum_s g_{is}\, g''_{is}}{\sqrt{\sum_i \sum_s g_{is}^2}\,\sqrt{\sum_i \sum_s (g''_{is})^2}}$$

where

$$\begin{aligned} g &= \text{present gradient} \\ g'' &= \text{previous gradient.} \end{aligned}$$

The initial gradient is set to a constant:

$$\text{Initial } g_{is} = \sqrt{\frac{1}{t}}$$

e) ***COSAV.*** Average cosine of the angle between successive gradients. This is a weighted average. For the first iteration, its value is set to 0.

$$\text{COSAV}_{\text{present}} = \text{CAGRGL}_{\text{present}} \times \text{COSAVW} + \text{COSAV}_{\text{previous}} \times (1.0 - \text{COSAVW})$$

where COSAVW is a weighting factor under the control of the user.

f) ***ACSAV.*** Average absolute value of the cosine of the angle between successive gradients. This is a weighted average. For the first iteration, its value is set to 0.

$$\text{ACSAV}_{\text{present}} = |\text{CAGRGL}_{\text{present}}| \times \text{ACSAVW} + \text{ACSAV}_{\text{previous}} \times (1.0 - \text{ACSAVW})$$

where ACSAVW is a weighting factor under the control of the user.

g) ***SFGR.*** Scale factor of the gradient. As the computation proceeds, the scale factor of successive gradients decreases. One way that the scaling procedure can stop is by reaching a user-supplied minimum value of the scale factor of the gradient.

$$\text{SFGR} = \sqrt{\frac{1}{n}\sum_i \sum_s g_{is}^2}$$

where g is the present gradient.

h) ***STEP.*** Step size. In the step size formula, the two main determinants of the new step size are the previous step size and angle factor. The step sizes used do not affect the final solution but they do affect the number of iterations required to reach a solution.

$$\text{STEP}_{\text{present}} = \text{STEP}_{\text{previous}} \times \text{angle factor} \times \text{relaxation factor} \times \text{good luck factor}$$

where

$$\begin{aligned} \text{angle factor} &= 4.0^{\text{COSAV}} \\ \text{relaxation (or bias) factor} &= \frac{1.4}{A\,B} \\ A &= 1 + (\min(1, \text{SRATAV}))^5 \\ B &= 1 + \text{ACSAV} - |\text{COSAV}| \\ \text{good luck factor} &= \sqrt{\min(1, \text{SRAT})} \end{aligned}$$

The first step size is computed as follows:

$$\text{STEP} = 50. \times \text{Stress} \times \text{SFGR}$$

48.5 Stress for Final Configuration

This is a reiteration of the last value of the Stress column of the history of computation (see 4.a above). Here the Stress is a measure of how well the final configuration matches the input data.

Interpretation of the stress for the final configuration depends on the formula used in the calculations. Note that the use of Stress SQDEV yields to substantially larger values of stress for the same degree of "goodness of fit".

For the classical mode of using MDSCAL, Kruskal and Carmone give the following table for the usual range of values of N (say from 10 to 30) and the usual range of dimensionality (say from 2 to 5):

	Stress SQDIST	Stress SQDEV
Poor	20.0 %	40.0 %
Fair	10.0 %	20.0 %
Good	5.0 %	10.0 %
Excellent	2.5 %	5.0 %
"Perfect"	0.0 %	0.0 %

48.6 Final Configuration

On each iteration the next configuration is formed by starting from the old configuration and moving along the (negative) gradient of stress a distance equal to the step size.

$$\text{New configuration} = \text{old configuration} + \frac{\text{STEP}}{\text{SFGR}}\,(\text{gradient})$$

Each row of the final configuration matrix provides the coordinates of one variable of the configuration. The orientation of the reference axes is arbitrary and thus one should look for rotated or even oblique axes that may be readily interpretable. If an ordinary Euclidean distance was used, it is possible to rotate the configuration so that its principal axes coincide with the coordinate axes. The CONFIG program can be used for this purpose.

48.7 Sorted Configuration

This is the final configuration presented with each dimension sorted - the coordinates are reordered from small to big.

48.8 Summary

a) ***IPOINT, JPOINT.*** These are variable subscripts, (i, j), indicating to which pair of variables refer the three statistics below.

b) ***DATA.*** For each variable pair, it is the input index of similarity or dissimilarity as provided by the user in the input data matrix.

c) ***DIST.*** This is the distance between points in the final configuration.

For Minkowski r - metric,

$$d_{ij} = \left[\sum_s |x_{is} - x_{js}|^r\right]^{1/r}$$

In the case of $r = 2$ it becomes an ordinary Euclidean distance

$$d_{ij} = \sqrt{\sum_s (x_{is} - x_{js})^2}$$

In the case of $r = 1$ it becomes a City block distance

$$d_{ij} = \sum_s |x_{is} - x_{js}|$$

d) ***DHAT.*** D-hats are the numbers which minimize the stress, subject to the constraint that the d-hats have the same rank order as the input data; they are "appropriate" distances, estimated from the input data.

They are obtained from

$$\sum_i \sum_j \widehat{d}_{ij} = \sum_i \sum_j d_{ij} \quad \text{and} \quad \widehat{d}_{ij} \geq \widehat{d}_{lm} \quad \text{if} \quad \begin{array}{ll} p_{ij} \leq p_{lm} & \text{(similarities)} \\ \text{or} & \\ p_{ij} \geq p_{lm} & \text{(dissimilarities)} \end{array}$$

where

d_{ij} = distance between variables i and j in the configuration
$\widehat{d}_{ij}$ = a monotonic transformation of the p_{ij}'s
p_{ij} = the input index of similarity or dissimilarity between variables i and j.

48.9 Note on Ties in the Input Data

Ties in the input data, i.e. identical values in the input data matrix, can be treated in either of two ways - the choice is up to the user.

The primary approach, DIFFER, treats ties in the input matrix as an indeterminate order relation, which can be resolved arbitrarily so as to decrease dimensionality or stress.

The secondary approach, EQUAL, treats ties as implying an equivalence relation, which (insofar as possible) is to be maintained (even if stress is increased).

If there are few ties, it does not make much difference which approach is chosen.

48.10 Note on Weights

The program provides for weighting, but it is not weighting in the usual IDAMS sense.

MDSCAL weighting may be used to assign differing importance to differing data values, that is, to assign weights to cells of the input data matrix. This sort of weighting can be used, for instance, to accommodate differing measurement variability among the data values.

If weights are used,

$$\text{Stress SQDIST} = \sqrt{\frac{\sum_i \sum_j w_{ij} \, (d_{ij} - \widehat{d}_{ij})^2}{\sum_i \sum_j w_{ij} \, d_{ij}^2}}$$

$$\text{Stress SQDEV} = \sqrt{\frac{\sum_i \sum_j w_{ij} \, (d_{ij} - \widehat{d}_{ij})^2}{\sum_i \sum_j w_{ij} \, (d_{ij} - \overline{d}\,)^2}}$$

where

$$\overline{d} = \frac{\sum_i \sum_j w_{ij} \, d_{ij}}{\sum_i \sum_j w_{ij}}$$

and w_{ij} indicates the value in the cell ij of the weight matrix.

48.11 References

Kruskal, J.B., Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis, *Psychometrica*, 3, 1964.

Kruskal, J.B., Nonmetric multidimensional scaling: a numerical method, *Psychometrica*, 29, 1964.

Chapter 49

Multiple Classification Analysis

Notation

y	=	value of the dependent variable
w	=	value of the weight
k	=	subscript for case
i	=	subscript for predictor
j	=	subscript for category within a predictor
p	=	number of predictors
c	=	number of non-empty categories across all predictors
a_{ij}	=	adjusted deviation of the j^{th} category of predictor i (see 2.c below)
N_{ij}	=	number of cases in the j^{th} category of predictor i
N	=	total number of cases
W	=	total sum of weights

subscript ijk indicates that the case k belongs to the j^{th} category of the predictor i.

49.1 Dependent Variable Statistics

a) ***Mean.*** Grand mean of y.

$$\overline{y} = \frac{\sum_k w_k\, y_k}{W}$$

b) ***Standard deviation of y (estimated).***

$$\widehat{s}_y = \sqrt{\left(\frac{N}{N-1}\right)\left[\frac{W\sum_k w_k\, y_k^2 - \left(\sum_k w_k\, y_k\right)^2}{W^2}\right]}$$

c) ***Coefficient of variation.***

$$C_y = \frac{100\,\widehat{s}_y}{\overline{y}}$$

d) ***Sum of y.***

$$\text{Sum of } y = \sum_k w_k\, y_k$$

e) ***Sum of y squared.***

$$\text{Sum of } y^2 = \sum_k w_k \, y_k^2$$

f) ***Total sum of squares.***

$$\text{TSS} = \sum_k w_k \, (y_k - \overline{y})^2$$

g) ***Explained sum of squares.***

$$\text{ESS} = \sum_i \sum_j a_{ij} \left(\sum_k w_{ijk} \, y_{ijk} \right)$$

h) ***Residual sum of squares.***

$$\text{RSS} = \text{TSS} - \text{ESS}$$

49.2 Predictor Statistics for Multiple Classification Analysis

a) ***Class mean.*** Mean of the dependent variable for cases in the j^{th} category of predictor i.

$$\overline{y}_{ij} = \frac{\sum_k w_{ijk} \, y_{ijk}}{\sum_k w_{ijk}}$$

b) ***Unadjusted deviation from grand mean.***

$$\text{Unadjusted } a_{ij} = \overline{y}_{ij} - \overline{y}$$

c) ***Coefficient.*** Adjusted deviation a_{ij} from grand mean. This is the regression coefficient for each category of each predictor.

$$\text{Predicted } y_k = \overline{y} + \sum_i a_{ijk}$$

The values of a_{ij} are obtained by an iterative procedure which stops when $\sum_k (y_k - predictedy_k)^2$ reaches the minimum.

d) ***Adjusted class mean.*** This is an estimate of what the mean would have been if the group had been exactly like the total population in its distribution over all the other predictor classifications. If there were no correlation among predictors, the adjusted mean would equal the class mean.

$$\text{Adjusted } \overline{y}_{ij} = \overline{y} + a_{ij}$$

e) ***Standard deviation (estimated)*** of the dependent variable for the j^{th} category of the predictor i.

$$\widehat{s}_{ij} = \sqrt{\frac{\sum_k w_{ijk} \, y_{ijk}^2 - \left(\sum_k w_{ijk} \, y_{ijk} \right)^2 / \sum_k w_{ijk}}{\sum_k w_{ijk} - \left(\sum_k w_{ijk} / \, N_{ij} \right)}}$$

f) ***Coefficient of variation (C.var.).***

$$C_{ij} = \frac{100 \, \widehat{s}_{ij}}{\overline{y}_{ij}}$$

g) ***Unadjusted deviation SS.*** This is the sum of squares of unadjusted deviations for predictor i.

$$U_i = \sum_j \Big(\sum_k w_{ijk} \Big) \left(\overline{y}_{ij} - \overline{y} \right)^2$$

h) ***Adjusted deviation SS.*** This is the sum of squares of adjusted deviations for predictor i.

$$D_i = \sum_j \Big(\sum_k w_{ijk} \Big) \left(a_{ij}^2 \right)$$

i) ***Eta squared for predictor i.*** Eta squared can be interpreted as the percent of variance in the dependent variable that can be explained by predictor i all by itself.

$$\eta_i^2 = \frac{U_i}{\text{TSS}}$$

j) ***Eta for predictor i.*** It indicates the ability of the predictor, using the categories given, to explain variation in the dependent variable.

$$\eta_i = \sqrt{\eta_i^2}$$

k) ***Eta squared for predictor i, adjusted for degrees of freedom.***

$$\text{Adjusted } \eta_i^2 = 1 - A \left(1 - \eta_i^2 \right)$$

where A is the adjustment for degrees of freedom (see 3.b below).

l) ***Eta for predictor i, adjusted.***

$$\text{Adjusted } \eta_i = \sqrt{1 - A \left(1 - \eta_i^2 \right)}$$

m) ***Beta squared for predictor i.*** Beta squared is the sum of squares attributable to the predictor, after "holding all other predictors constant", relative to the total sum of squares. This is *not* in terms of percent of variance explained.

$$\beta_i^2 = \frac{D_i}{\text{TSS}}$$

n) ***Beta for predictor i.*** Beta provides a measure of ability of the predictor to explain variation in the dependent variable after adjusting for the effect of all other predictors. Beta coefficients indicate the relative importance of the various predictors (the higher the value the more variation is explained by the corresponding beta).

$$\beta_i = \sqrt{\beta_i^2}$$

49.3 Analysis Statistics for Multiple Classification Analysis

a) ***Multiple R squared unadjusted.*** This is the multiple correlation coefficient squared. It indicates the actual proportion of variance explained by the predictors used in the analysis.

$$R^2 = \frac{\text{ESS}}{\text{TSS}}$$

b) ***Adjustment for degrees of freedom.***

$$A = \frac{N - 1}{N - p - c - 1}$$

c) ***Multiple R squared adjusted.*** It provides an estimate of the multiple correlation in the population from which the sample was drawn. Note that it is an estimate of the multiple correlation which would be obtained if the same predictors, but not necessarily the same coefficients, were used for the population.

$$\text{Adjusted } R^2 = 1 - A\,(1 - R^2)$$

d) ***Multiple R adjusted.*** This is the multiple correlation coefficient adjusted for degrees of freedom. It is an estimate of the R which would be obtained if the same predictors were applied to the population.

$$\text{Adjusted } R = \sqrt{1 - A\,(1 - R^2)}$$

49.4 Summary Statistics of Residuals

The residual for a case k is $r_k = y_k - predictedy_k$,

a) ***Mean.***

$$\overline{r} = \frac{\sum_k w_k r_k}{W}$$

b) ***Variance (estimated).***

$$\widehat{s}_r^2 = \left(\frac{N}{N-1}\right)\left[\frac{W\sum_k w_k r_k^2 - \left(\sum_k w_k r_k\right)^2}{W^2}\right]$$

c) ***Skewness.*** The skewness of the distribution of residuals is measured by

$$g_1 = \left(\frac{N}{N-2}\right)\left(\frac{m_3}{\widehat{s}_r^2\sqrt{\widehat{s}_r^2}}\right)$$

where

$$m_3 = \frac{\sum_k w_k\,(r_k - \overline{r})^3}{W}$$

d) ***Kurtosis.*** The kurtosis of the distribution of residuals is measured by

$$g_2 = \left(\frac{N}{N-3}\right)\left(\frac{m_4}{(\widehat{s}_r^2)^2}\right) - 3$$

where

$$m_4 = \frac{\sum_k w_k\,(r_k - \overline{r})^4}{W}$$

49.5 Predictor Category Statistics for One-Way Analysis of Variance

See “One-Way Analysis of Variance” chapter for details.

49.6 One-Way Analysis of Variance Statistics

See "One-Way Analysis of Variance" chapter for details. Note that the adjustment factor A used in MCA program for one-way analysis of variance is calculated differently than in ONEWAY program, namely:

$$A = \frac{N-1}{N-c}$$

49.7 References

Andrews, F.M., Morgan, J.N., Sonquist, J.A., and Klem, L., *Multiple Classification Analysis*, 2nd ed., Institute for Social Research, The University of Michigan, Ann Arbor, 1973.

Chapter 50

Multivariate Analysis of Variance

Notation

y	=	value of dependent variable or covariate
i, j	=	subscripts for categories of predictors
k	=	subscript for case
p	=	number of dependent variables
df_h	=	degrees of freedom for the hypothesis
df_e	=	degrees of freedom for error.

50.1 General Statistics

a) ***Cell means.*** Let y_{ijk} represent a value of a dependent variable or covariate for the k^{th} case in the i, j^{th} subclass of a two-way classification.

$$\overline{y}_{ij} = \frac{\sum_{k=1}^{N_{ij}} y_{ijk}}{N_{ij}}$$

where N_{ij} is equal to the number of cases in the i, j^{th} subclass.

b) ***Basis of design.*** The design matrix is generated by first developing for each factor a one-way design matrix (a one-way K_f matrix) in accordance with the contrast type specified by the user for that factor. The overall design matrix K is obtained from the one-way K_f matrices by taking the Kronecker product of the matrices.

The design matrix is always printed with the effects equations in columns, beginning with the grand mean effect in the first column.

c) ***Intercorrelations among the normal equations coefficients.*** The basis of design is weighted by the cell counts. The effect of unequal cell frequencies is to introduce correlations between columns of the design matrix. These are those correlations. If the cell frequencies are equal, there will be 1's on the diagonal and zeros elsewhere.

d) ***Solution of the normal equations.*** The parameters are estimated by least squares in the form

$$LX = (K'DK)^{-1} K'DY$$

where

L = the contrast matrix which has as rows i the independent contrasts

in the parameters which are to be estimated and tested

X = the parameters to be estimated

K = the design matrix

D = a diagonal matrix with the number of cases in each cell

Y = a matrix of cell means with columns corresponding to variables.

When dealing with an orthogonal design and orthogonal contrasts, the contrasts have independent estimates. For unequal cell frequencies, however, the K appropriate for orthogonal designs is no longer orthogonal. It is required to transform K to orthogonality in the metric D. This is done by putting

$$T = SK'D^{1/2} \quad \text{with} \quad TT' = T'T = I = SK'DKS'$$

so

$$K'D^{1/2} = S^{-1}T$$

and

$$(K'DK)^{-1} = S'S$$

and, substituting in the first equation above,

$$(S')^{-1}LX = SK'DY$$

This last equation defines a new set of parameters which are linear functions of the contrasts, with the matrix SK' replacing K'. These parameters are orthogonal.

S is the matrix which produces the Gram-Schmidt orthogonalization of K in the metric D and reduces the rows of this to unit length. S, and thus $(S')^{-1}$, is triangular.

e) ***Partitioning of matrices.*** In a univariate analysis of variance, each case has one dependent variable y; in a multivariate analysis of variance, each case has a vector y of dependent variables. The multivariate analogue of y^2 is the matrix product $y'y$ and the multivariate analogue of a sum of squares is a sum of matrix products.

In a multivariate analysis, there is a matrix corresponding to each sum of squares in a univariate design. Multivariate tests depend on partitions of the total sum of products just as univariate tests depend on partitions of the total sum of squares. The formulas for the total sum of products, the between-subclasses sum of products, and the within-subclasses sum of products are

$$S_t = Y'Y$$

$$S_b = Y.'DY.$$

$$S_w = Y'Y - Y.'DY.$$

where

Y = the original $N \times p$ data matrix (N cases, p dependent variables)

$Y.$ = the $n \times p$ matrix of cell means (n cells, p dependent variables)

D = a diagonal matrix with the number of cases in each cell.

The between-subclasses sum of products is partitioned further according to the effects in the model.

f) ***Error correlation matrix.*** In a multivariate analysis of variance, the error term is a variance-covariance matrix. This is that error term reduced to a correlation matrix.

The correlation matrix is calculated using S_w, the within, or error, sum or products.

$$R_e = s_e^{-1}\, S_w\, s_e^{-1}$$

where

S_w = the within-subclasses sum of products
s_e^2 = the diagonal entries of S_w.

R_e is the matrix of correlation coefficients among the variates which estimate population values.

If the user specified that the within-subclasses sum of squares was to be augmented to form the error term, augmentation takes place before the matrix is reduced to correlations.

g) ***Principal components of the error correlation matrix.*** This is a standard principal components analysis of the matrix R_e. It indicates the factor structure of the variables found in the population under study. The eigenvalues (or roots) are printed beneath the components.

h) ***Error dispersion matrix.*** This is the error term, a variance-covariance matrix, for the analysis. The matrix is adjusted for covariates, if any. Each diagonal element of the matrix is exactly what would appear in a conventional analysis of variance table as the within mean square error for the variable.

$$M_e = \frac{S_w}{df_e}$$

where

S_w = the within-subclasses sum of products
df_e = the degrees of freedom for error, adjusted for augmentation if that was requested.

If augmentation is not requested, the degrees of freedom for error equals the number of cases minus the number of cells in the design.

i) ***Standard errors of estimation.*** They correspond to the square roots of the diagonal elements of the matrix M_e.

50.2 Calculations for One Test in a Multivariate Analysis

The calculations are repeated for each test requested by the user. Results of internal calculations described below under points a) to d) are not printed.

a) ***Sum of squares matrix due to hypothesis.*** The between-subclasses sum of squares is partitioned according to the various effects in the model. For a given hypothesis to be tested, the program determines the orthogonal estimates to be tested and computes the sum of squares due to hypothesis (S_h).

b) S_w ***and*** S_h ***reduced to mean squares and scaled to correlation space.*** The mean square matrix for the hypothesis, M_h, is calculated analogously to the means squares for error.

$$M_h = \frac{S_h}{df_h}$$

where

S_h = the sum of squares matrix due to hypothesis (see above).

The degrees of freedom for the hypothesis depend on the test requested; for a test of main effect A, where factor A has "a" levels, the degrees of freedom for hypothesis would be $a - 1$.

M_h is a matrix of the between-subclass mean products associated with a main effect or interaction hypothesis.

Both M_e and M_h are scaled to correlation space:

$$R_e = \Delta_e^{-1} M_e \Delta_e^{-1}$$

$$C_h = \Delta_e^{-1} M_h \Delta_e^{-1}$$

where

R_e = the matrix of correlation coefficients among the variables estimating population values
C_h = a matrix, which, although not a correlation matrix, does present the variances and covariances for the variables as affected by the treatment
M_e = the mean squares for error
M_h = the mean squares for hypothesis
Δ_e = a diagonal matrix containing the standard errors of estimation.

The matrix R_e is computed twice, once as described in the section "Error correlation matrix" and once as descibed here. If no covariates were specified, the results are identical and the second R_e matrix is not printed. If one or more covariates was specified, the second R_e matrix incorporates adjustements for the covariate(s).

c) ***Solution of the determinental equation.*** The usual method of computing Wilk's likelihood ratio criterion is from the determinental equation

$$|M_h - \lambda M_e| = 0$$

The above equation is pre-and-post-multiplied by the diagonal matrix Δ_e^{-1}

$$|\Delta_e^{-1} M_h \Delta_e^{-1} - \lambda R_e| = 0$$

Let

$$R_e = FF'$$

where

F = the matrix of principal components coefficients satisfying $F'F = \omega$, the diagonal matrix of eigenvalues of R_e.

Second determinental equation is pre-multiplied by F^{-1} and post-multiplied by its transpose giving

$$|(\Delta_e F)^{-1} M_h ((\Delta_e F)^{-1})' - \lambda F^{-1}(FF')(F^{-1})'| = 0$$

or

$$|(\Delta_e F)^{-1} M_h ((\Delta_e F)^{-1})' - \lambda I| = 0$$

The last equation is then solved for the values λ.

d) ***Likelihood ratio criterion.***

$$\Lambda = \prod_{q=1}^{s} \left(1 + \frac{df_h}{df_e} \times \lambda_q\right)^{-1}$$

where

λ_q = the non-zero values from the last equation in the previous section.

e) ***F-ratio for likelihood ratio criterion.*** The program uses the F-approximation to the percentage points of the null distribution of Λ.

$$F = \frac{1 - \Lambda^{1/k}}{\Lambda^{1/k}} \times \frac{k(2df_e + df_h - p - 1) - p(df_h) + 2}{2p(df_h)}$$

where

$$k = \sqrt{\frac{p^2(df_h)^2 - 4}{p^2 + (df_h)^2 - 5}}$$

This is a multivariate test of significance of the effect for all the dependent variables simultaneously.

f) ***Degrees of freedom for the F-ratio.***

$$p(df_h)$$

and

$$\frac{k(2df_e + df_h - p - 1) - p(df_h) + 2}{2}$$

If $p = 1$ or 2 and $df_h = 1$ or 2, k is set to 1 in cases when $p(df_h) = 2$.

g) ***Canonical variances of the principal components of the hypothesis.*** These are the lambdas calculated as described in the section "Solution of the determinental equation" above. They are ordered by decreasing magnitude. The number of non-zero lambdas for a given equation is equal to df_h (the number of degrees of freedom associated with M_h), or p, the number of dependent variables, whichever is smaller.

h) ***Coefficients of the principal components of the hypothesis.*** Solving equation

$$|(\Delta_e F)^{-1} M_h ((\Delta_e F)^{-1})' - \lambda I| = 0$$

gives rise to T, for which

$$F^{-1}\ \Delta_e^{-1}\ M_h\ \Delta_e^{-1}(F^{-1})' = T\,\lambda\,T'$$

This can be rewritten as

$$T'\ F^{-1}\ \Delta_e^{-1}\ X_h\ X_h'\ \Delta_e^{-1}(F^{-1})'T = \lambda$$

The above equation is considered as

$$T'\ F^{-1}\ \Delta_e^{-1}\ X_h = S_h^*$$

where

$$S_h^*\ (S_h^*)' = \lambda$$

and written in usual factor equation form, $X = FS$, is

$$\Delta_e^{-1}\ X_h = FTS_h^*$$

The coefficients of the principal components of the hypothesis, FT, are printed by the program.

i) ***Contrast component scores for estimated effects.*** The rows of S_h^* are the sets of factor scores, atributable to hypothesis that have as maximum variances the λ_i.

j) ***Cumulative Bartlett's tests on the roots.*** The tests can be used to determine the dimensionality of the configuration. The lambdas, or roots, are ordered in ascending order of magnitude. In the Bartlett's tests, all the roots are tested first. Then all but the first, then all but the first two, and so forth. The Chi-square test provides a test of the significance of the variance accounted for by the $n-k$ roots after the acceptance of the first k roots.

First the lambdas are scaled

$$\text{normed } \lambda_i = \frac{df_h}{df_e} \times \lambda_i$$

and then **Chi-square** is calculated

$$\chi^2_{k+1} = \left(df_e + df_h - \frac{df_h + p + 1}{2} \right) \left(\sum_{i=k+1}^{s} \ln(\text{normed } \lambda_i + 1) \right)$$

where

$$\begin{aligned} k &= \text{the number of accepted roots } \ (k = 0, 1, ..., s-1) \\ s &= \text{the number of roots.} \end{aligned}$$

The degrees of freedom are

$$DF = (p-k)(g-k-1)$$

where g is equal to the number of levels of the hypothesis.

k) ***F-ratios for univariate tests.*** These are the diagonal elements of $\Delta_e^{-1} M_h \Delta_e^{-1}$. The F-ratio for variable y is exactly the F-ratio which would be obtained for the given effect if a univariate analysis were performed with variable y being the only dependent variable.

50.3 Univariate Analysis

If a single dependent variable is specified, the calculations are nonetheless performed as outlined above. Advantage, however, is taken of simplification, e.g. the principal component of the error correlation "matrix" is set equal to one and no calculation is done.

Result of a univariate analysis of variance is a conventional ANOVA table with small differences. It contains a row for grand mean but does not contain a row for the total. The grand mean is generally not interpretable. To obtain the total sum of squares, sum all the sums of squares except the sum for the grand mean.

50.4 Covariance Analysis

The formulas and discussion above do not, for the most part, take into account covariates. If one or more covariates was specified, it is the sums of products matrices, S_e and S_h which are adjusted. If there are q covariates, the program begins by carrying them along with p dependent variables. There is a $(p \times q) \times (p \times q)$ sum of product of error, S_e matrix, and $(p \times q) \times (p \times q)$ S_h matrix for each hypothesis. The total matrix S_t is computed. S_e and S_h are partitioned into sections corresponding to the dependent variables and covariates. Reduced $(p \times p)$ error and total matrices are obtained and reduced matrix for hypothesis is then obtained by subtraction.

Error correlation matrix and the principal components of this matrix are computed after the adjustment to S_e for covariates.

Chapter 51

One-Way Analysis of Variance

Notation

y	=	value of the dependent variable
w	=	value of the weight
k	=	subscript for case
i	=	subscript for category of the control variable
N_i	=	number of cases in category i
W_i	=	sum of weights for category i
N	=	total number of cases
W	=	total sum of weights
c	=	number of code categories of the control variable with non-zero degrees of freedom.

51.1 Descriptive Statistics for Categories of the Control Variable

a) ***Mean.***

$$\overline{y}_i = \frac{\sum_k w_{ik}\, y_{ik}}{W_i}$$

b) ***Standard deviation (estimated).***

$$\widehat{s}_i = \sqrt{\left(\frac{N_i}{N_i - 1}\right)\left[\frac{W_i \sum_k w_{ik}\, y_{ik}^2 - \left(\sum_k w_{ik}\, y_{ik}\right)^2}{W_i^2}\right]}$$

c) ***Coefficient of variation (C.var.).***

$$C_i = \frac{100\ \widehat{s}_i}{\overline{y}_i}$$

d) ***Sum of y.***

$$\text{Sum } y_i = \sum_k w_{ik}\, y_{ik}$$

e) ***Percent.***

$$\text{Percent}_i = \frac{\text{Sum } y_i}{\sum_i \text{Sum } y_i}$$

f) ***Sum of y squared.***

$$\text{Sum } y_i^2 = \sum_k w_{ik}\, y_{ik}^2$$

g) ***Total.*** The total row gives the statistics 1.a through 1.e above computed over all cases, except those in code categories with zero degrees of freedom.

h) ***Degrees of freedom for the category i.***

$$\text{df}_i = W_i\,(N_i - 1)\,/\,N_i$$

Categories with zero degrees of freedom are not included in the computation of summary statistics.

51.2 Analysis of Variance Statistics

a) ***Total sum of squares.***

$$\text{TSS} = \sum_i \sum_k w_{ik}\, y_{ik}^2 - \frac{\left(\sum_i \sum_k w_{ik}\, y_{ik}\right)^2}{W}$$

b) ***Between means sum of squares.*** This is sometimes called the between groups (or inter-groups) sum of squares.

$$\text{BSS} = \sum_i \left[\frac{\left(\sum_k w_{ik}\, y_{ik}\right)^2}{\sum_k w_{ik}}\right] - \frac{\left(\sum_i \sum_k w_{ik}\, y_{ik}\right)^2}{W}$$

c) ***Within groups sum of squares.*** This is sometimes called the intra-groups sum of squares.

$$\text{WSS} = \text{TSS} - \text{BSS}$$

d) ***Eta squared.*** This measure can be interpreted as the percent of variance in the dependent variable that can be explained by the control variable. It ranges from 0 to 1.

$$\eta^2 = \frac{\text{BSS}}{\text{TSS}}$$

e) ***Eta.*** This is a measure of the strength of the association between the dependent variable and the control variable. It ranges from 0 to 1.

$$\eta = \sqrt{\frac{\text{BSS}}{\text{TSS}}}$$

f) ***Eta squared adjusted.*** Eta squared adjusted for degrees of freedom.

$$\text{Adjusted } \eta^2 = 1 - A(1 - \eta^2)$$

with adjustment factor

$$A = \frac{W - 1}{W - c}$$

g) ***Eta adjusted.***

$$\text{Adjusted } \eta = \sqrt{\text{Adjusted } \eta^2}$$

h) ***F value.*** The F ratio can be referred to the F distribution with $c - 1$ and $N - c$ degrees of freedom. A significant F ratio means that mean differences, or effects, probably exist among the groups.

$$F = \frac{\text{BSS}/(c - 1)}{\text{WSS}/(N - c)}$$

The F ratio is not computed if a weight variable was specified.

Chapter 52

Partial Order Scoring

52.1 Special Terminology and Definitions

Let denote a set of elements by $\mathcal{V} = \{a, b, c, \dots, \}$ and a binary relation defined on it by R.

a) ***Binary relation.*** A binary relation R in $\mathcal{V}$ is such that for any two elements $a, b \in \mathcal{V}$

$$aRb$$

For every binary relation R in $\mathcal{V}$ there exists a *converse* relation R^+ in $\mathcal{V}$ such that

$$bR^+a$$

b) ***Reflexive and anti-reflexive relation.*** A relation R is reflexive when

$$aRa \qquad \text{for all } a \in \mathcal{V}$$

and R is anti-reflexive when

$$not(aRa) \qquad \text{for all } a \in \mathcal{V}$$

c) ***Symmetric and anti-symmetric relation.*** A relation R is symmetric when $R = R^+$, that is when

$$aRb \Longleftrightarrow bRa \qquad \text{for all } a, b \in \mathcal{V}$$

and R is anti-symmetric when symmetry does not appear for all $a \neq b$.

d) ***Transitive relation.*** A relation R is transitive when

$$aRb \wedge bRc \Longrightarrow aRc \qquad \text{for all } a, b, c \in \mathcal{V}$$

e) ***Equivalence relation.*** A relation R defined on a set of elements $\mathcal{V}$ is an equivalence relation when it is:

- reflexive,
- symmetric, and
- transitive.

Note that the commonly used "equality" relation, (=), defined on the set of real numbers is an equivalence relation.

f) ***Strict partial order relation.*** A relation R is called a strict partial order when it satisfies the conditions:

- aRb and bRa cannot hold simultaneously, and

- R is transitive.

A strict partial order relation is denoted hereafter by $\prec$.

g) ***Partially ordered set.*** A set $\mathcal{V}$ is called a partially ordered set if a strict partial order relation "$\prec$" is defined on it. The fundamental properties of a partially ordered set are:

- $a \prec b \wedge b \prec c \Longrightarrow a \prec c \qquad$ for all $a, b, c \in \mathcal{V}$
- $a \prec b$ and $b \prec a$ cannot hold simultaneously.

h) ***Ordered set.*** A set $\mathcal{V}$ is called an ordered set if there are two relations "$\approx$" and "$\prec$" defined on it and they satisfy the axioms of ordering:

- for any two elements $a, b \in \mathcal{V}$, one and only one of the relations $a \approx b$, $a \prec b$, $b \prec a$ holds,
- "$\approx$" is an equivalence relation, and
- "$\prec$" is a transitive relation.

In other words, an ordered set is a partially ordered set with additional equivalence relation defined on it, and where the conditions "neither $a \prec b$ nor $b \prec a$" and "$a \approx b$" are equivalent.

i) ***Subset of elements dominating an element a.***

$$\mathcal{G}(a) = \Big\{\, g \mid g \in \mathcal{V};\ a \prec g \,\Big\}$$

j) ***Subset of elements dominated by an element a.***

$$\mathcal{L}(a) = \Big\{\, l \mid l \in \mathcal{V};\ l \prec a \,\Big\}$$

k) ***Subset of comparable elements.***

$$\mathcal{C}(a) = \mathcal{G}(a) \cup \mathcal{L}(a)$$

Note that $\mathcal{G}(a) \cap \mathcal{L}(a) = \emptyset$.

l) ***Strict dominance.*** An element b strictly dominates an element a if

$$a \prec b \quad \text{and} \quad \text{not}(b \prec a)$$

It can also be said that "b is strictly better than a", or that "a is strictly worse than b".

52.2 Calculation of Scores

Let denote a list of variables to be used in the analysis by

$$\{x_1, x_2, \ldots, x_i, \ldots, x_v\}$$

and a priority list associated to them by

$$\{p_1, p_2, \ldots, p_i, \ldots, p_v\}.$$

The PARTIAL ORDER RELATION constructed on the basis of this collection of variables,

$$a \prec b \quad \text{for any cases } a \text{ and } b$$

is equivalent to the condition

$$x_1(a) \le x_1(b),\ x_2(a) \le x_2(b),\ \ldots,\ x_v(a) \le x_v(b)$$

where $x_i(a)$ and $x_i(b)$ denote values of the i^{th} variable for cases a and b respectively.

When COMPARING TWO CASES, the variables of highest priority (lowest LEVEL value) are considered first. If they unambiguously determine the relation, the comparison procedure ends. In the situation of equality,

the comparison is continued using variables of the next priority level. This procedure is repeated until the relation is determined at one of the priority levels, or the end of the variable list is reached.

For each case a from the analyzed set, the program calculates:

$\overline{N}(a)$ = the number of cases strictly dominating the case a
$N(a)$ = the number of cases equivalent to the case a
$\underline{N}(a)$ = the number of cases strictly dominated by the case a

and then one (or two) of the following scores:

$$s_1(a) = S \frac{\underline{N}(a)}{\underline{N}(a) + N(a) + \overline{N}(a)}$$

$$r_1(a) = S - s_1(a)$$

$$s_2(a) = S \frac{\underline{N}(a) + N(a)}{\underline{N}(a) + N(a) + \overline{N}(a)}$$

$$r_2(a) = S - s_2(a)$$

$$s_3(a) = S \frac{\underline{N}(a)}{N}$$

$$r_3(a) = S \frac{\overline{N}(a) + N(a)}{N}$$

$$s_4(a) = S \frac{\underline{N}(a) + N(a)}{N}$$

$$r_4(a) = S \frac{\overline{N}(a)}{N}$$

where

N = total number of cases in the analyzed set
S = the value of the scale factor (see the SCALE parameter).

The values of the ORDER parameter select the score(s) as follows:

ASEA : $r_3(a)$
DEEA : $s_4(a)$
ASCA : $r_4(a)$
DESA : $s_3(a)$
ASER : $s_1(a), r_1(a)$
DESR : $s_1(a), r_1(a)$
ASCR : $s_2(a), r_2(a)$
DEER : $s_2(a), r_2(a)$.

52.3 References

Debreu, G., Representation of a preference ordering by a numerical function, *Decision Process*, eds. R.M. Thrall, C.A. Coombs and R.L. Davis, New York, 1954.

Hunya, P., *A Ranking Procedure Based on Partially Ordered Sets*, Internal paper, JATE, Szeged, 1976.

Chapter 53

Pearsonian Correlation

Notation

x, y	=	values of variables
w	=	value of the weight
k	=	subscript for case
N	=	number of valid cases on both x and y
W	=	total sum of weights.

53.1 Paired Statistics

They are computed for variables taken by pair (x, y) on the subset of cases having valid data on both x and y.

a) ***Adjusted weighted sum.*** The number of cases, weighted, with valid data on both x and y.

b) ***Mean of x.***

$$\overline{x} = \frac{\sum_k w_k\, x_k}{W}$$

Note: the formula for *mean of y* is analogous.

c) ***Standard deviation of x (estimated).***

$$\widehat{s}_x = \sqrt{\left(\frac{N}{N-1}\right)\left[\frac{W\sum_k w_k\, x_k^2 - \left(\sum_k w_k\, x_k\right)^2}{W^2}\right]}$$

Note: the formula for *standard deviation of y* is analogous.

d) ***Correlation coefficient.*** Pearson's product moment coefficient r.

$$r_{xy} = \frac{W\sum_k w_k\, x_k\, y_k - \left(\sum_k w_k\, x_k\right)\left(\sum_k w_k\, y_k\right)}{\sqrt{\left[W\sum_k w_k\, x_k^2 - \left(\sum_k w_k\, x_k\right)^2\right]\left[W\sum_k w_k\, y_k^2 - \left(\sum_k w_k\, y_k\right)^2\right]}}$$

e) ***t-test.*** This statistic is used to test the hypothesis that the population correlation coefficient is zero.

$$t = \frac{r\sqrt{N-2}}{\sqrt{1-r^2}}$$

53.2 Unpaired Means and Standard Deviations

They are computed variable by variable for all variables included in the analysis using the formulas given in 1.a, 1.b and 1.c respectively, the potential difference in results being due to different number of valid cases.

a) ***Adjusted weighted sum.*** The number of cases, weighted, with valid data on x.

b) ***Mean of x.*** Mean of variable x for all cases with valid data on x.

c) ***Standard deviation of x (estimated).*** Standard deviation of variable x for all cases with valid data on x.

53.3 Regression Equation for Raw Scores

It is computed on all valid cases for the pair (x, y).

a) ***Regression coefficient.*** This is the unstandardized regression coefficient of y (dependent variable) on x (independent variable).

$$B_{yx} = r_{xy}\left(\frac{\widehat{s}_y}{\widehat{s}_x}\right)$$

b) ***Constant term.***

$$A = \overline{y} - B_{yx}\,\overline{x}; \qquad \text{regression equation:} \quad y = B_{yx}\,x + A$$

53.4 Correlation Matrix

The elements of this matrix are computed on the basis of the formula given under 1.d above. Note that standard deviations output with correlation matrix are calculated according to the formula given under 1.c above (estimated standard deviations).

53.5 Cross-products Matrix

It is a square matrix with the following elements:

$$\mathrm{CP}_{xy} = \sum_k w_k\,x_k\,y_k$$

53.6 Covariance Matrix

It is a matrix containing the following elements:

$$\mathrm{COV}_{xy} = r_{xy}\,s_x\,s_y$$

where

$$s_x = \sqrt{\frac{W\sum_k w_k\,x_k^2 - \left(\sum_k w_k\,x_k\right)^2}{W^2}}$$

and s_y is calculated according to the analogous formula.

Note that the covariance matrix output by PEARSON does not contain diagonal elements. In order to allow their recalculation, standard deviations output with this matrix are calculated according to the above formula (unestimated standard deviations).

Chapter 54

Rank-ordering of Alternatives

Notation

i, j, l	=	subscripts for alternatives
m	=	number of alternatives
k	=	case index
n	=	number of cases
w	=	value of the weight.

54.1 Handling of Input Data

Let a SET OF ALTERNATIVES be denoted by $\mathcal{A} = \{a_1, a_2, \ldots, a_i, \ldots, a_m\}$ and the set of sources of information (called hereafter EVALUATIONS) be denoted by $\mathcal{E} = \{e_1, e_2, \ldots, e_k, \ldots, e_n\}$.

In practice, data providing the primary information on the preference relations may appear in rather various forms. The program accepts, however, two basic types of data: data representing a selection of alternatives and data representing a ranking of alternatives. All other forms of data should be transformed by the user prior to the execution of the RANK program.

a) ***Data representing a selection of alternatives.*** In this case the evaluations represent the choice of the mostly preferred alternatives and optionally their preference order. In other words, all the evaluations e_k select a subset $\mathcal{A}_k$ from $\mathcal{A}$ and optionally order the elements of it. For this reason $\mathcal{A}_k$ is a subset of alternatives (ordered or non-ordered), and the $\mathcal{A}_k$'s constitute the primary individual data:

$$\mathcal{A}_k = \left\{ a_{ki_1}, a_{ki_2}, \ldots, a_{ki_{p_k}} \right\}$$

where

p	=	maximum number of alternatives which could be selected in an evaluation
p_k	=	number of alternatives actually selected in the evaluation e_k

and $p_k \leq p < m$.

b) ***Data representing a ranking of alternatives.*** Here the evaluations represent the ranking of the alternatives within the whole set $\mathcal{A}$, and the attribution to each of them of its rank number. Formally, all the evaluations e_k give a rank number $\rho_k(a_i) = \rho_{ki}$ to all the alternatives. In this case the data are provided in the following form:

$$\mathcal{P}_k = \{\rho_k(a_1), \rho_k(a_2), \ldots, \rho_k(a_m)\}$$

Note that an alternative a_{ki_1} "is strictly preferred to" or "strictly dominates" another alternative a_{ki_2} according to the data coming from the evaluation e_k if the former has a rank higher than the latter.

Similarly, an alternative a_{ki_1} "is preferred to" or "dominates" another alternative a_{ki_2} according to the data coming from the evaluation e_k if the rank of a_{ki_1} is at least as high as the rank of a_{ki_2}. The value "1" is taken for the highest rank.

Only the data described in paragraph b) are directly processed by the program. The data depicted in a) are transformed into the form of b). This transformation makes a distinction between the strict and the weak preference.

The TRANSFORMATION RULE, when dealing with data representing a completely ordered selection of alternatives (strict preference), is the following:

$$\text{for } a_i \in \mathcal{A}_k \qquad \rho_k(a_{i_1}) = 1,\ \rho_k(a_{i_2}) = 2,\ \ldots,\ \rho_k(a_{ip_k}) = p_k$$
$$\text{for } a_i \notin \mathcal{A}_k \qquad \rho_k(a_i) = \frac{p_k + 1 + m}{2}$$

When dealing with data representing a non-ordered selection of alternatives (weak preference), it is assumed that all the selected alternatives are at the same level of preference. According to this assumption, the transformation rule is:

$$\text{for } a_i \in \mathcal{A}_k \qquad \rho_k(a_i) = \frac{p_k + 1}{2}$$
$$\text{for } a_i \notin \mathcal{A}_k \qquad \rho_k(a_i) = \frac{p_k + 1 + m}{2}$$

As a result of the transformations defined above, the preference (or priority choice) data are for the next steps of analyses in the form:

$$\mathcal{P}_{(n,m)} = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1i} & \cdots & \rho_{1m} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2i} & \cdots & \rho_{2m} \\ \vdots & \vdots & & \vdots & & \vdots \\ \rho_{k1} & \rho_{k2} & \cdots & \rho_{ki} & \cdots & \rho_{km} \\ \vdots & \vdots & & \vdots & & \vdots \\ \rho_{n1} & \rho_{n2} & \cdots & \rho_{ni} & \cdots & \rho_{nm} \end{bmatrix}$$

54.2 Method of Classical Logic Ranking

In this method the matrix $\mathcal{P}$ is used as the initial data for the analysis. Concerning the strict or weak character of the preference relation it should be noted that it plays a role only in the steps leading to the matrix $\mathcal{P}$. In the further steps of the analysis, the procedure is controlled by other parameters, such as rank difference for concordance and rank difference for discordance (see below).

The classical logic ranking procedure consists of two major steps, namely: a) construction of the relations, and b) identification of cores.

a) ***Construction of the relations.*** In this step, two "working" relations (the concordance relation and the discordance relation) are constructed first. Then they are used to construct a final dominance relation.

i) THE CONCORDANCE AND DISCORDANCE RELATIONS are build from the matrix $\mathcal{P}_{(n,m)}$, and the rules applied in this process are essentially the same for both relations.

CONCORDANCE RELATION. Two parameters are used in creating a relation which reflects the concordance of the collective opinion that "a_i is preferred to a_j":

d_c = the rank difference for concordance $(0 \le d_c \le m-1)$

p_c = the minimum proportion for concordance $(0 \le p_c < 1)$.

Rank difference for concordance enables the user to influence the evaluation of data when constructing the individual preference matrices

$$\mathrm{RC}^k(d_c) = \left[\mathrm{rc}^k_{ij}(d_c) \right] \quad \text{where } i, j = 1, 2, \ldots, m.$$

The elements of $\mathrm{RC}^k(d_c)$, which measure the dominance of a_i over a_j according to the evaluation k, are defined as follows:

$$\mathrm{rc}_{ij}^k(d_c) = \begin{cases} 1 & \text{if } \rho_{kj} - \rho_{ki} \geq d_c \\ 0 & \text{otherwise.} \end{cases}$$

The aggregation of these matrices measures the average dominance of a_i over a_j and has the form of a fuzzy relation described by the matrix

$$\mathrm{RC}(d_c) = \left[\mathrm{rc}_{ij}(d_c) \right]$$

where

$$\mathrm{rc}_{ij}(d_c) = \frac{\sum_k w_k \, \mathrm{rc}_{ij}^k(d_c)}{\sum_k w_k}$$

Note that higher d_c values lead to more rigorous construction rules, since $d_c^1 < d_c^2$ implies

$$\mathrm{rc}_{ij}^k(d_c^1) \geq \mathrm{rc}_{ij}^k(d_c^2) \quad \text{and} \quad \mathrm{rc}_{ij}(d_c^1) \geq \mathrm{rc}_{ij}(d_c^2)$$

Minimum proportion for concordance makes it possible to transform the fuzzy relation $\mathrm{RC}(d_c)$ into a non-fuzzy one, called the concordance relation, described by the matrix

$$\mathrm{RC}(d_c, p_c) = \left[\mathrm{rc}_{ij}(d_c, p_c) \right]$$

the elements of which are defined as follows:

$$\mathrm{rc}_{ij}(d_c, p_c) = \begin{cases} 1 & \text{if } \mathrm{rc}_{ij}(d_c) \geq p_c \\ 0 & \text{otherwise.} \end{cases}$$

The condition $\mathrm{rc}_{ij}(d_c, p_c) = 1$ means that the collective opinion is in concordance with the statement "a_i is preferred to a_j" at the level (d_c, p_c).

It is clear again that increasing the p_c value one obtains stricter conditions for the concordance.

Discordance relation. The construction of the discordance relation follows the same way as was explained for the concordance. The two parameters controlling the construction are:

$$\begin{aligned} d_d &= \text{the rank difference for discordance } \ (0 \leq d_d \leq m-1) \\ p_d &= \text{the maximum proportion for discordance } \ (0 \leq p_d \leq 1). \end{aligned}$$

The individual discordance relations are determined first in the matrices

$$\mathrm{RD}^k(d_d) = \left[\mathrm{rd}_{ij}^k(d_d) \right] \quad \text{where } \ i, j = 1, 2, \ldots, m.$$

The elements of $\mathrm{RD}^k(d_d)$, which measure the dominance of a_j over a_i according to the evaluation k, are defined as follows:

$$\mathrm{rd}_{ij}^k(d_d) = \begin{cases} 1 & \text{if } \rho_{ki} - \rho_{kj} \geq d_d \\ 0 & \text{otherwise.} \end{cases}$$

The aggregation of these matrices measures the average dominance of a_j over a_i and has the form of a fuzzy relation described by the matrix

$$\mathrm{RD}(d_d) = \left[\mathrm{rd}_{ij}(d_d) \right]$$

where

$$\mathrm{rd}_{ij}(d_d) = \frac{\sum_k w_k \, \mathrm{rd}_{ij}^k(d_d)}{\sum_k w_k}$$

As for concordance, the second parameter (maximum proportion for discordance), enables the user to transform the fuzzy relation $\mathrm{RD}(d_d)$ into a non-fuzzy one, called the discordance relation, described by the matrix

$$\mathrm{RD}(d_d, p_d) = \left[\mathrm{rd}_{ij}(d_d, p_d) \right]$$

the elements of which are defined as follows:

$$\mathrm{rd}_{ij}(d_d, p_d) = \begin{cases} 1 & \text{if } \mathrm{rd}_{ij}(d_d) > p_d \\ 0 & \text{otherwise.} \end{cases}$$

The condition $\mathrm{rd}_{ij}(d_d, p_d) = 1$ means that the collective opinion is in discordance with the statement "a_i is preferred to a_j", i.e. supports the opposite statement "a_j is preferred to a_i", at the level (d_d, p_d). This can be interpreted as a "collective veto" against the statement "a_i is preferred to a_j".

Note that higher values of d_d and p_d lead to less rigorous construction rules and thus to weaker conditions for discordance.

ii) THE DOMINANCE RELATION is composed of the concordance and discordance relations. The basic idea is that the statement "a_i is preferred to a_j" can be accepted if the collective opinion

- is in concordance with it, i.e. $\mathrm{rc}_{ij}(d_c, p_c) = 1$, and
- is not in discordance with it, i.e. $\mathrm{rd}_{ij}(d_d, p_d) = 0$;

otherwise this statement has to be rejected. So the dominance relation, being a function of four parameters, is described by the matrix $\mathcal{R}$ of $m \times m$ dimensions

$$\mathcal{R} = \Big[\, r_{ij}(d_c, p_c, d_d, p_d) \,\Big]$$

where the elements are obtained according to the expression

$$r_{ij}(d_c, p_c, d_d, p_d) = \min\big[\, \mathrm{rc}_{ij}(d_c, p_c),\ 1 - \mathrm{rd}_{ij}(d_d, p_d) \,\big]$$

The r_{ij} is a monotonously decreasing function of the first two parameters, and a monotonously increasing function of the last two ones. This implies that:

- by increasing the d_c, p_c and/or decreasing d_d, p_d one can diminish the number of connections in the dominance relation, and
- by changing the parameters in the opposite direction one can create more connections.

b) ***Identification of cores.*** The CORES are subsets of $\mathcal{A}$ (set of alternatives) consisting of non-dominated alternatives. An alternative a_j is non-dominated if and only if

$$r_{ij} = 0 \quad \text{for all } i = 1, 2, \ldots, m.$$

i) According to this criterion the core of the set $\mathcal{A}$ (the highest level core) is the subset

$$\mathcal{C}(\mathcal{A}) = \Big\{\, a_j \mid a_j \in \mathcal{A};\ r_{ij} = 0,\ i = 1, 2, \ldots, m \,\Big\}$$

- If $\mathcal{C}(\mathcal{A}) = \emptyset$ then all the alternatives are dominated.
- If $\mathcal{C}(\mathcal{A}) = \mathcal{A}$ then all the alternatives are non-dominated.

ii) In order to find the subsequent core, the elements of the previous core are removed from the dominance relation first. This means that the corresponding rows and columns are removed from the relational matrix. Then the search for a new core is repeated in the reduced structure.

The successive application of i) and ii) gives a series of cores $\mathcal{A}_1^c$, $\mathcal{A}_2^c$, ..., $\mathcal{A}_q^c$. These cores represent consecutive layers of alternatives with decreasing ranks in the preference structure, while the alternatives belonging to the same core are assumed to be of the same rank.

54.3 Methods of Fuzzy Logic Ranking: the Input Relation

In the fuzzy logic ranking methods, the matrix $\mathcal{P}_{(n,m)}$ is used to construct: a) individual preference relations, and b) the input relation (called also "a fuzzy relation") on the set of alternatives $\mathcal{A}$. Here the strict and weak character of the preference relation plays an important role.

a) ***Construction of individual preference relations.*** For each evaluation e_k an individual preference relation, which is given implicitly in $\mathcal{P}$, is transformed into the matrix of $m \times m$ dimensions:

$$\mathcal{R}^k = \Big[\, r_{ij}^k \,\Big] \quad \text{where } i, j = 1, 2, \ldots, m$$

in which

$$r_{ij}^k = \begin{cases} 1 & \text{if the statement "}a_i \text{ is preferred to } a_j \text{ in the evaluation } e_k\text{" is true;} \\ 0 & \text{if this statement is false.} \end{cases}$$

Depending on the preference type used, the statement "a_i is preferred to a_j in the evaluation e_k" is equivalent to the inequality

$$\begin{array}{ll} \rho_{ki} < \rho_{kj} & \text{(strict preference), or} \\ \rho_{ki} \leq \rho_{kj} & \text{(weak preference).} \end{array}$$

b) ***Construction of the input relation (fuzzy relation).*** The aggregation of the individual preference relation matrices provides the matrix representing a fuzzy relation on the set of alternatives $\mathcal{A}$:

$$\mathcal{R} = \left[r_{ij} \right]$$

where

$$r_{ij} = \frac{\sum_k w_k \, r_{ij}^k}{\sum_k w_k}$$

Each component r_{ij} of $\mathcal{R}$ can be interpreted as the credibility of the statements "a_i is preferred to a_j" in a global sense, and without referring to the single evaluation. Thus, the following general interpretation is possible:

$$\begin{array}{ll} r_{ij} = 1 & \text{"}a_i \text{ is preferred to } a_j\text{" in all the evaluations,} \\ r_{ij} = 0 & \text{"}a_i \text{ is preferred to } a_j\text{" in no evaluation,} \\ 0 < r_{ij} < 1 & \text{"}a_i \text{ is preferred to } a_j\text{" in a certain portion of the evaluations.} \end{array}$$

c) ***Characteristics of the input relation.***

i) Fuzzyness

$$\begin{array}{ll} \text{non-fuzzy :} & \text{if } r_{ij} = 0 \text{ or } r_{ij} = 1 \text{ for all } \; i, j = 1, 2, \ldots, m; \\ \text{fuzzy :} & \text{otherwise.} \end{array}$$

ii) Symmetry

$$\begin{array}{ll} \text{symmetric :} & \text{if } r_{ij} = r_{ji} \text{ for all } \; i, j = 1, 2, \ldots, m; \\ \text{anti-symmetric :} & \text{if } r_{ij} \neq 0 \text{ implies } r_{ji} = 0 \text{ for all } \; i \neq j; \\ \text{asymmetric :} & \text{otherwise.} \end{array}$$

iii) Reflexivity

$$\begin{array}{ll} \text{reflexive :} & \text{if } r_{ii} = 1 \text{ for all } \; i = 1, 2, \ldots, m; \\ \text{anti-reflexive :} & \text{if } r_{ii} = 0 \text{ for all } \; i = 1, 2, \ldots, m; \\ \text{irreflexive :} & \text{otherwise.} \end{array}$$

iv) Trichotomy

$$\begin{array}{ll} \text{trichotome :} & \text{if } r_{ij} + r_{ji} = 1 \text{ for all } \; i, j = 1, 2, \ldots, m \text{ and } i \neq j; \\ \text{(normalized)} & \\ \text{non-trichotome :} & \text{otherwise.} \\ \text{(non-normalized)} & \end{array}$$

v) Coherence index. Its value, C, depends on the order of the rows and columns in $\mathcal{R}$, i.e. on the order of the alternatives in $\mathcal{A}$, and $-1 \leq C \leq 1$.

$$C = \frac{\sum_{i<j} (r_{ij} - r_{ji})}{\sum_{i<j} (r_{ij} + r_{ji})}$$

ABSOLUTE COHERENCE INDEX is an order-independent modification of C. Its value, C_a, is the upper bound for C and $0 \le C_a \le 1$.

$$C_a = \frac{\sum_{i<j} |r_{ij} - r_{ji}|}{\sum_{i<j} (r_{ij} + r_{ji})}$$

Indices C and C_a are indicators of unanimity in the preference data. A full coherence is shown when $C = 1$, while $C_a = 0$ indicates a full lack of coherence. The value -1 of the index C can be interpreted as an order of alternatives opposite to the order defined by the fuzzy relation.

vi) INTENSITY INDEX. This index can be interpreted as an average credibility level of the statements "a_i is preferred to a_j" or "a_j is preferred to a_i". In general, its value $-1 \le I \le 2$, while in the case of a strict preference $0 \le I \le 1$. Here $I = 1$ implies a normalized relation (see 3.c below) and means that in all the preference data one of the above statements is valid for all the pairs of alternatives.

$$I = \frac{\sum_{i<j} (r_{ij} + r_{ji})}{m(m-1)/2}$$

vii) DOMINANCE INDEX. It is also an order-dependent index, and $-1 \le D \le 1$.

$$D = \frac{\sum_{i<j} (r_{ij} - r_{ji})}{m(m-1)/2}$$

ABSOLUTE DOMINANCE INDEX, similarly to the coherence index, is defined as the order independent dominance index. Its value, D_a, is the upper bound for D and $0 \le D_a \le 1$.

$$D_a = \frac{\sum_{i<j} |r_{ij} - r_{ji}|}{m(m-1)/2}$$

The indices D and D_a indicate the average difference between the credibility of the statements "a_i is preferred to a_j" and of their opposite statements "a_j is preferred to a_i" .

Note that C, I, D and C_a, I, D_a are not independent of one another, namely:

$$C \cdot I = D \quad \text{and} \quad C_a \cdot I = D_a$$

d) ***Normalized matrix.*** A normalized matrix is obtained from the $\mathcal{R}$ matrix using the following transformation:

$$r'_{ij} = \begin{cases} \dfrac{r_{ij}}{r_{ij} + r_{ji}} & \text{if } i \ne j \text{ and } r_{ij} + r_{ji} \ne 0 \\ r_{ij} & \text{otherwise.} \end{cases}$$

54.4 Fuzzy Method-1: Non-dominated Layers

The fuzzy logic ranking methods assume a fuzzy preference relation with the membership function $\mu : \mathcal{A} \times \mathcal{A} \longrightarrow [0, 1]$ on a given set $\mathcal{A}$ of alternatives. This membership function is represented by the matrix $\mathcal{R}$ (see section 3 above). The values $r_{ij} = \mu(a_i, a_j)$ are understood as the degrees to which the preferences expressed by the statements "a_i is preferred to a_j" are true.

Another assumption is that:

in the case of weak preference, μ is reflexive, i.e.

$$\mu(a_i, a_i) = r_{ii} = 1 \qquad \text{for all } a_i \in \mathcal{A}$$

in the case of strict preference, μ is anti-reflexive, i.e.

$$\mu(a_i, a_i) = r_{ii} = 0 \qquad \text{for all } a_i \in \mathcal{A}$$

The fuzzy method-1 procedure looks for A SET OF NON-DOMINATED ALTERNATIVES (denoted ND alternatives), considering such a set as the highest level core of alternatives. The reason for this is that ND

alternatives are either equivalent to one another, or are not comparable to one another on the basis of the preference relation considered, and they are not dominated in a strict sense by others.

In order to determine a fuzzy set of ND alternatives, two fuzzy relations corresponding to the given preference relation $\mathcal{R}$ are defined: fuzzy quasi-equivalence relation and fuzzy strict preference relation. Formally they are defined as follows:

fuzzy quasi-equivalence relation $\mathcal{R}^e$:

$$\mathcal{R}^e = \mathcal{R} \cap \mathcal{R}^{-1}$$

fuzzy strict preference relation $\mathcal{R}^s$:

$$\mathcal{R}^s = \mathcal{R} \setminus \mathcal{R}^e = \mathcal{R} \setminus (\mathcal{R} \cap \mathcal{R}^{-1}) = \mathcal{R} \setminus \mathcal{R}^{-1}$$

where $\mathcal{R}^{-1}$ is a relation opposite to the relation $\mathcal{R}$.

Furthermore, the following membership functions are defined respectively for $\mathcal{R}^e$ and $\mathcal{R}^s$:

$$\mu^e(a_i, a_j) = \min(r_{ij}, r_{ji})$$

$$\mu^s(a_i, a_j) = \begin{cases} r_{ij} - r_{ji} & \text{when } r_{ij} > r_{ji} \\ 0 & \text{otherwise.} \end{cases}$$

For any fixed alternative $a_j \in \mathcal{A}$ the function $\mu^s(a_j, a_i)$ describes a fuzzy set of alternatives which are strictly dominated by a_j. The complement of this fuzzy set, described by the membership function $1 - \mu^s(a_j, a_i)$, is for any fixed a_j the fuzzy set of all the alternatives which are not strictly dominated by a_j. Then the intersection of all such complement fuzzy sets (over all $a_j \in \mathcal{A}$) represents the fuzzy set of those alternatives $a_i \in \mathcal{A}$ which are not strictly dominated by any of the alternatives from the set $\mathcal{A}$. This set is called the fuzzy set μ^{ND} of ND alternatives in the set $\mathcal{A}$. Thus, according to the definition of intersection

$$\mu^{\mathrm{ND}}(a_i) = \min_{a_j \in \mathcal{A}} (1 - \mu^s(a_j, a_i)) = 1 - \max_{a_j \in \mathcal{A}} \mu^s(a_j, a_i)$$

The value $\mu^{\mathrm{ND}}(a_i)$ represents the degree to which the alternative a_i is not strictly dominated by any of the alternatives from the set $\mathcal{A}$.

The HIGHEST LEVEL CORE OF ALTERNATIVES contains those alternatives a_i which have the greatest degree of non-dominance or, in other words, which give a value for $\mu^{\mathrm{ND}}(a_i)$ that is equal to the value:

$$M^{\mathrm{ND}} = \max_{a_i \in \mathcal{A}} \mu^{\mathrm{ND}}(a_i)$$

The value of M^{ND} is called THE CERTAINTY LEVEL corresponding to the core defined by:

$$\mathcal{C}(\mathcal{A}) = \left\{ a_i \,|\, a_i \in \mathcal{A};\ \ \mu^{\mathrm{ND}}(a_i) = M^{\mathrm{ND}} \right\}$$

The subsequent cores are constructed by a repeated application of the procedure described above. The elements of the previous core are removed from the fuzzy relation first, i.e. the corresponding rows and columns are removed from the fuzzy relation matrix. Then the calculations are repeated in the reduced structure.

54.5 Fuzzy Method-2: Ranks

The input relation to this method is the same as to the method-1, namely: the matrix $\mathcal{R}$ which has to be reflexive or anti-reflexive. However, the question to be answered here is quite different.

The fuzzy method-2 procedure looks for the LEVEL OF CREDIBILITY, denoted c_{jp}, OF STATEMENTS "a_j is exactly at the p^{th} place in the ordered sequence of the alternatives in $\mathcal{A}$", denoted T_{jp}. The c_{jp} values form a matrix $\mathcal{M}$ of $m \times m$ dimensions representing a fuzzy membership function, in which the rows correspond to the alternatives and the columns to the possible positions in the sequence $1, 2, \ldots, m$.

In order to make possible the calculation of c_{jp}'s they must be decomposed into already known credibility levels r_{ij}, and thus the statements T_{jp} must be decomposed into elementary statements with known credibility levels r_{ij}. For that, further notations are introduced. Note that for an alternative a_j being exactly at the p^{th} place means that it is preferred to $m - p$ alternatives and is preceded by the remaining $p - 1$

alternatives. When the subset of alternatives after a_j is fixed, then

$$\begin{array}{lcl} \mathcal{A}^j_{m-p} & = & \text{the subset of those alternatives to which } a_j \text{ is preferred,} \\ \mathcal{A}^j_{p-1} & = & \text{the subset of alternatives which are preferred to } a_j, \\ \mathcal{A}^j & = & \text{the subset } \mathcal{A} \setminus \{a_j\}. \end{array}$$

Obviously,

$$\mathcal{A}^j_{p-1} \cup \mathcal{A}^j_{m-p} = \mathcal{A}^j$$

$$\mathcal{A}^j_{p-1} \cap \mathcal{A}^j_{m-p} = \emptyset$$

and the statement T_{jp} is equivalent to a sequence of statements "a_j is preferred to all the elements of $\mathcal{A}^j_{m-p}$ and all the elements of $\mathcal{A}^j_{p-1}$ are preferred to a_j", connected by the disjunctive operator of logic.

Furthermore, the statement "a_j is preferred to all the elements of $\mathcal{A}^j_{m-p}$" is a conjunction of the already known statements "a_j is preferred to a_l", with the credibility level equal to r_{jl}, for all the elements a_l of $\mathcal{A}^j_{m-p}$.

Similarly, the statement "all the elements of $\mathcal{A}^j_{p-1}$ are preferred to a_j" is a conjunction of the already known statements "a_i is preferred to a_j", with the credibility level equal to r_{ij}, for all the elements a_i of $\mathcal{A}^j_{m-p}$.

Applying the corresponding fuzzy operators, the elements of the matrix $\mathcal{M}$ can be obtained as follows:

$$c_{jp} = \max_{\mathcal{A}^j_{m-p} \subseteq \mathcal{A}^j} \left[\min \Big(\min_{a_l \in \mathcal{A}^j_{m-p}} r_{jl}, \min_{a_i \in \mathcal{A}^j_{p-1}} r_{ij} \Big) \right]$$

The computation of the c_{jp} values is performed using an optimization procedure which produces a series of subsets $\mathcal{A}^j_{m-p}$ (while keeping j and p fixed) with strictly monotonously increasing values of the function to be maximized in successive steps.

The program provides two ways of interpretation of the matrix $\mathcal{M}$.

FUZZY SETS OF RANKS BY ALTERNATIVES.

For each alternative a_j, a fuzzy membership function values show the credibility of having this alternative at the p^{th} place ($p = 1, 2, \ldots, m$). Also, the most credible ranks (places) for each alternative are listed.

FUZZY SUBSETS OF ALTERNATIVES BY RANKS.

For each rank (place) p, a fuzzy membership function value shows the credibility of the alternative a_j ($j = 1, 2, \ldots, m$) to be at this place. Also the most credible alternatives, candidates for the place, are listed.

54.6 References

Dussaix, A.-M., *Deux méthodes de détermination de priorités ou de choix*, Partie 1: *Fondements mathématiques*, Document UNESCO/NS/ROU/624, UNESCO, Paris, 1984.

Jacquet-Lagrèze, E., Analyse d'opinions valuées et graphes de préférence, *Mathématiques et sciences humaines*, 33, 1971.

Jacquet-Lagrèze, E., L'agrégation des opinions individuelles, *Informatique et sciences humaines*, 4, 1969.

Kaufmann, A., *Introduction à la théorie des sous-ensembles flous*, Masson, Paris, 1975.

Orlovski, S.A., Decision-making with a fuzzy preference relation, *Fuzzy Sets and Systems*, Vol.1, No 3, 1978.

Chapter 55

Scatter Diagrams

Notation

x	=	value of the variable to be plotted horizontally
y	=	value of the variable to be plotted vertically
w	=	value of the weight
k	=	subscript for case
N	=	total number of cases
W	=	total sum of weights.

55.1 Univariate Statistics

These unweighted statistics are calculated for all variables used in the execution.

a) ***Mean.***

$$\overline{x} = \frac{\sum_k x_k}{N}$$

b) ***Standard deviation.***

$$s_x = \sqrt{\frac{\sum_k x_k^2}{N} - \overline{x}^2}$$

55.2 Paired Univariate Statistics

They are calculated on the set of cases having valid data on both x and y. These are weighted statistics if a weight variable is specified.

a) ***Mean.***

$$\overline{x} = \frac{\sum_k w_k\, x_k}{W}$$

Note: the formula for $\overline{y}$ is analogous.

b) ***Standard deviation.***

$$s_x = \sqrt{\frac{\sum\limits_k w_k\, x_k^2}{W} - \overline{x}^2}$$

Note: the formula for s_y is analogous.

c) ***N.*** The number of cases, weighted, with valid data on both x and y.

55.3 Bivariate Statistics

They are calculated on the set of cases having valid data on both x and y.

a) ***Pearson's product moment r.***

$$r_{xy} = \frac{W \sum\limits_k w_k\, x_k\, y_k - \left(\sum\limits_k w_k\, x_k\right)\left(\sum\limits_k w_k\, y_k\right)}{\sqrt{\left[W \sum\limits_k w_k\, x_k^2 - \left(\sum\limits_k w_k\, x_k\right)^2\right]\left[W \sum\limits_k w_k\, y_k^2 - \left(\sum\limits_k w_k\, y_k\right)^2\right]}}$$

b) ***Regression statistics: constant A and coefficient B.***

$$A = \frac{\sum\limits_k w_k\, y_k - \sum\limits_k w_k\, x_k\, B}{W}$$

where B is the unstandardized regression coefficient.

$$B = \frac{W \sum\limits_k w_k\, x_k\, y_k - \left(\sum\limits_k w_k\, x_k\right)\left(\sum\limits_k w_k\, y_k\right)}{W \sum\limits_k w_k\, x_k^2 - \left(\sum\limits_k w_k\, x_k\right)^2}$$

The constant A and coefficient B can be used in the regression equation $y = Bx + A$ to predict y from x.

Chapter 56

Searching for Structure

Notation

y	=	value of the dependent variable
x	=	frequency (weighted) of the categorical dependent variable or values (weighted) of dichotomous dependent variables
z	=	value of the covariate
w	=	value of the weight
k	=	subscript for case
j	=	subscript for category code of the dependent variable or subscript for dichotomous dependent variables
m	=	number of codes of the dependent variable or number of dichotomous dependent variables
g	=	subscript for group; $g = 1$ indicate the whole sample
i	=	subscript for final groups
t	=	number of final groups
N_g	=	number of cases in group g
W_g	=	sum of weights in group g
N_i	=	number of cases in the final group i
W_i	=	sum of weights in the final group i
N	=	total number of cases
W	=	total sum of weights.

56.1 Means analysis

This method can be used when analysing one dependent variable (interval or dichotomous) and several predictors. It aims at creating groups which would allow for the best prediction of the dependent variable values from the group average. In other words, created groups should provide largest differences in group means. Thus, the splitting criterion (explained variation) is based upon group means.

a) ***Trace statistics.*** These are the statistics calculated on the whole sample (for $g = 1$), and on tentative splits for parent groups as well as for each group resulting from the best split.

i) SUM (WT). Number of cases (N_g) if the weight variable is not specified, or weighted number of cases (W_g) in group g.

ii) MEAN Y. Mean value of the dependent variable y in group g.

$$\overline{y}_g = \frac{\sum_{k=1}^{N_g} w_k \, y_{gk}}{W_g}$$

iii) VAR Y. Variance of the dependent variable y in group g.

$$\sigma^2_{y_g} = \frac{\sum_{k=1}^{N_g} w_k \, (y_{gk} - \overline{y}_g)^2}{W_g - \frac{W_g}{N_g}}$$

iv) VARIATION. Sum of squares of the dependent variable (as in one-way analysis of variance) in group g.

$$V_g = \sum_{k=1}^{N_g} w_k \, (y_{gk} - \overline{y}_g)^2$$

v) VAR EXPL. Explained variation is measured by the difference between the variation in the parent group and the sum of variation in the two children groups. It provides, for each predictor, the amount of variation explained by the best split for this predictor, i.e. the highest value obtained over all possible splits for this predictor.

Let g_1 and g_2 denote two subgroups (children groups) obtained in a split of the parent group g, and V_{g_1} and V_{g_2} their respective variation. The variation explained by such a split of group g is calculated as follows:

$$EV_g = V_g \; - \; (V_{g_1} \; + \; V_{g_2})$$

Then, this value is maximized over all possible splits for the predictor.

vi) EXPLAINED VARIATION. This is the percent of the total variation explained by the final groups.

$$Percent = 100 \, \frac{EV}{TV}$$

where EV and TV are, respectively, the variation explained by the final groups and the total variation (see 1.b below).

b) ***One-way analysis of final groups.*** These are one-way analysis of variance statistics calculated for the final groups.

i) EXPLAINED VARIATION and DF. This is the amount of variation explained by the final groups and the corresponding degrees of freedom.

$$EV = TV - UV = TV - \sum_{i=1}^{t} V_i$$

$$DF = t - 1$$

ii) TOTAL VARIATION and DF. Variation calculated for the whole sample, i.e. for group 1, and the corresponding degrees of freedom.

$$TV = V_1$$

$$DF = W - 1$$

iii) ERROR and DF. This is the amount of unexplained variation and the corresponding degrees of freedom.

$$UV = \sum_{i=1}^{t} V_i$$

$$DF = W - t$$

c) ***Split summary table.*** The table provides group mean value, variance and variation of the dependent variable at each split as well as the variation explained by that split (see 1.a above).

d) ***Final group summary table.*** The table provides mean value, variance and variation of the dependent variable for the final groups (see 1.a above).

e) ***Percent of explained variation.*** The percent of total variation explained by the best split for each group is calculated as follows:

$$Percent_g = 100\ \frac{EV_g}{TV}$$

Note that this value is equal to zero for the final groups (indicated by an asterisk).

f) ***Residuals.*** The residuals are the differences between the observed value and the predicted value of the dependent variable.

$$e_k = y_k - \widehat{y}_k$$

As predicted value, a case is assigned the mean value of the dependent variable for the group to which it belongs, i.e.

$$\widehat{y}_{ik} = \overline{y}_i$$

56.2 Regression Analysis

This method can be used when analysing a dependent variable (interval or dichotomous) with one covariate and several predictors. It aims at creating groups which would allow for the best prediction of the dependent variable values from the group regression equation and the value of covariate. In other words, created groups should provide largest differences in group regression lines. The splitting criterion (explained variation) is based upon group regression of the dependent variable on the covariate.

a) ***Trace statistics.*** These are the statistics calculated on the whole sample (for $g = 1$), and on tentative splits for parent groups as well as for each group resulting from the best split.

i) SUM (WT). Number of cases (N_g) if the weight variable is not specified, or weighted number of cases (W_g) in group g.

ii) MEAN Y,Z. Mean value of the dependent variable y and the covariate z in group g (see 1.a.ii above).

iii) VAR Y,Z. Variance of the dependent variable y and the covariate z in group g (see 1.a.iii above).

iv) SLOPE. This is the slope of the dependent variable y on the covariate z in group g.

$$b_g = \frac{\sum_{k=1}^{N_g} w_k\,(y_{gk} - \overline{y}_g)(z_{gk} - \overline{z}_g)}{\sum_{k=1}^{N_g} w_k\,(z_{gk} - \overline{z}_g)^2}$$

v) VARIATION. This is the error or residual sum of squares from estimating the variable y by its regression on covariate in group g, i.e. a measure of deviation about the regression line.

$$V_g = \sum_{k=1}^{N_g} w_k\,(y_{gk} - \overline{y}_g)^2 \;-\; b_g \times \sum_{k=1}^{N_g} w_k\,(y_{gk} - \overline{y}_g)(z_{gk} - \overline{z}_g)$$

where b_g is the slope of the regression line in group g.

vi) VAR EXPL. Explained variation (EV). See 1.a.v above for general information, and 2.a.v above for details on V (variation) used in regression analysis.

vii) EXPLAINED VARIATION. This is the percent of the total variation explained by the final groups. See 1.a.vi above and 2.b below.

b) ***One-way analysis of final groups.*** These are the summary statistics for the final groups. See 1.b above for general information, and 2.a.v and 2.a.vi above for details on V and EV measures used in regression analysis.

c) ***Split summary table.*** The table provides group mean value, variance and variation of the dependent variable at each split as well as the variation explained by that split. It also provides mean value and variance of the covariate. See 2.a above for formulas. Moreover, the following regression statistics are calculated for each split:

i) SLOPE. It is the slope of the dependent variable y on the covariate z in group g (see 2.a.iv above).

ii) INTERCEPT. It is the constant term in the regression equation.

$$a_g = \overline{y}_g - b_g\,\overline{z}_g$$

where b_g is the slope in group g.

iii) CORR. Pearson r correlation coefficient between the dependent variable y and the covariate z in group g.

$$r_g = \frac{\sum_{k=1}^{N_g} w_k\,(y_{gk} - \overline{y}_g)\,(z_{gk} - \overline{z}_g)}{\sqrt{\sigma^2_{y_g}\,\sigma^2_{z_g}}}$$

d) ***Final group summary table.*** The table provides the same information (except the explained variation) as in "Split summary table", but for final groups.

e) ***Percent of explained variation.*** The percent of total variation explained by the best split for each group (see 1.e and 2.a.vi above).

f) ***Residuals.*** The residuals are the differences between the observed value and the predicted value of dependent variable.

$$e_k = y_k - \widehat{y}_k$$

Predicted values are calculated as follows:

$$\widehat{y}_{ik} = a_i + b_i\,z_{ik}$$

where a_i and b_i are regression coefficients for the final group i.

56.3 Chi-square Analysis

This method can be used when analysing one dependent variable (nominal or ordinal) or a set of dichotomous dependent variables with several predictors. It aims at creating groups which would allow for the best prediction of the dependent variable category from its group distribution. In other words, created groups should provide largest differences in the dependent variable distributions. The splitting criterion (explained variation) is calculated on the basis of frequency distributions of the dependent variable. Note that multiple dependent dichotomous variables are treated as categories of one categorical variable.

a) ***Trace statistics.*** These are the statistics calculated on the whole sample (for $g = 1$), and on tentative splits for parent groups as well as for each group resulting from the best split.

i) SUM (WT). Number of cases (N_g) if the weight variable is not specified, or weighted number of cases (W_g) in group g.

ii) VARIATION. This is the entropy for group g, i.e. a measure of disorder in the distribution of the dependent variable.

$$V_g = -2 \sum_{j=1}^{m} x_{jg\cdot} \times \ln \frac{x_{jg\cdot}}{x_{\cdot g\cdot}}$$

where

$$x_{jg\cdot} = \sum_{k=1}^{N_g} x_{jgk} \qquad x_{\cdot g\cdot} = \sum_{j=1}^{m} x_{jg\cdot}$$

and x_{jgk} is the "frequency" (coded 0 or 1) of code j (or value of variable j) of case k in group g.

iii) VAR EXPL. Explained variation (EV). See 1.a.v above for general information, and 3.a.ii above for details on V (variation) used in chi-square analysis.

iv) EXPLAINED VARIATION. This is the percent of the total variation explained by the final groups. See 1.a.vi above and 3.b below.

b) ***One-way analysis of final groups.*** These are the summary statistics for the final groups. See 1.b above for general information, and 3.a.ii and 3.a.iii above for details on V and EV measures used in chi-square analysis.

c) ***Split summary table.*** The table provides variation of the dependent variable at each split as well as the variation explained by that split. See 3.a.ii and 3.a.iii above for formulas.

d) ***Final group summary table.*** The table provides variation of the dependent variable for the final groups.

e) ***Percent of explained variation.*** The percent of total variation explained by the best split for each group (see 1.e and 3.a.iii above).

f) ***Percent distributions.*** A bivariate table showing percentage distributions of the dependent variable for all groups (P_{jg}).

g) ***Residuals.*** The residuals are the differences between the observed value and the predicted value of dependent variable.

For analysis with ONE CATEGORICAL DEPENDENT VARIABLE, residuals are calculated for each category of the variable. Thus, the number of residuals is equal to the number of categories.

$$e_{jk} = x_{jk} - \widehat{x}_{jik}$$

Observed values, x_{jk}, are created as a series of "dummy variables", coded 0 or 1.

As predicted value for category j, a case is assigned the proportion of cases being in this category for the group to which the case belongs, i.e.

$$\widehat{x}_{jik} = P_{ji}/100$$

For analysis with SEVERAL DICHOTOMOUS DEPENDENT VARIABLES, residuals are calculated for each variable. Thus, the number of residuals is equal to the number of dependent variables.

$$e_{jk} = x'_{jk} - \widehat{x}_{jik}$$

Observed values are calculated as follows:

$$x'_{jk} = \frac{x_{jk}}{\sum_{j=1}^{m} x_{jk}}$$

As predicted value for variable j, a case is assigned the proportion of cases having value 1 for this variable in the group to which the case belongs, i.e.

$$\widehat{x}_{jik} = P_{ji}/100$$

56.4 References

Morgan, J.N., Messenger, R.C., *THAID A Sequential Analysis Program for the Analysis of Nominal Scale Dependent Variables*, Institute for Social Research, The University of Michigan, Ann Arbor, 1973.

Sonquist, J.A., Baker, E.L., Morgan, J.N., *Searching for Structure*, Revised ed., Institute for Social Research, The University of Michigan, Ann Arbor, 1974.

Chapter 57

Univariate and Bivariate Tables

Notation

x	=	value of the row variable in bivariate tables, or value of the variable in univariate tables
y	=	value of the column variable in bivariate tables
w	=	value of the weight
k	=	subscript for case
i	=	subscript for row in bivariate tables
j	=	subscript for column in bivariate tables
r	=	number of rows in bivariate tables
c	=	number of columns in bivariate tables
$f_{i\cdot}$	=	marginal frequency in the row i of a bivariate table
$f_{\cdot j}$	=	marginal frequency in the column j of a bivariate table
N	=	total number of cases.

57.1 Univariate Statistics

a) ***Wtnum.*** The weight variable number, or zero if the weight variable is not specified.

b) ***Wtsum.*** Number of cases if the weight variable is not specified, or weighted number of cases (sum of weights).

c) ***Mode.*** The first category which contains the maximum frequency.

d) ***Median.*** The median is calculated as an n-tile with two requested subintervals. See "Distribution and Lorenz Functions" chapter for details.

e) ***Mean.***

$$\overline{x} = \frac{\sum_k w_k x_k}{\sum_k w_k}$$

f) ***Variance.*** This is an unbiased estimation of the population variance.

$$\widehat{s}_x^2 = \left(\frac{N}{N-1}\right) \frac{\sum_k w_k \left(x_k - \overline{x}\right)^2}{\sum_k w_k}$$

g) ***Standard deviation.*** It should be noted that $\widehat{s}_x$ is not itself an unbiased estimate of the population standard deviation.

$$\widehat{s}_x = \sqrt{\widehat{s}_x^2}$$

h) ***Coefficient of variation (C.var.).***

$$C_x = \frac{100\ \widehat{s}_x}{\overline{x}}$$

i) ***Skewness.*** The skewness of the distribution of x is measured by

$$g_1 = \left(\frac{N}{N-2}\right)\left(\frac{m_3}{\widehat{s}_x^2\sqrt{\widehat{s}_x^2}}\right) \quad \text{where} \quad m_3 = \frac{\sum\limits_k w_k\,(x_k - \overline{x})^3}{\sum\limits_k w_k}$$

Skewness is a measure of asymmetry. Distributions which are skewed to the right, i.e. the tail is on the right, have positive skewness; distributions which are skewed to the left have negative skewness; a normal distribution has skewness equal to 0.0.

j) ***Kurtosis.*** The kurtosis of the distribution of x is measured by

$$g_2 = \left(\frac{N}{N-3}\right)\left(\frac{m_4}{(\widehat{s}_x^2)^2}\right) - 3 \quad \text{where} \quad m_4 = \frac{\sum\limits_k w_k\,(x_k - \overline{x})^4}{\sum\limits_k w_k}$$

Kurtosis measures the peakedness of a distribution. A normal distribution has kurtosis equal to 0.0. A curve with a sharper peak has positive kurtosis; distributions less peaked than a normal distribution have negative kurtosis.

k) ***n-tiles.*** The n-tile break points are calculated the same way as in the QUANTILE program.

57.2 Bivariate Statistics

a) ***Chi-square.*** Chi-square is appropriate for testing the significance of differences of distributions among independent groups.

$$\chi^2 = \sum_i \sum_j \frac{(f_{ij} - E_{ij})^2}{E_{ij}}$$

where

f_{ij} = the observed frequency in cell ij

E_{ij} = the expected(calculated) frequency in cell ij;
it is the product of the frequency of the row i times
the frequency in the column j, divided by the total N.

For two by two tables, the χ^2 is computed according to the following formula:

$$\chi^2 = \frac{N(|ad - bc| - N/2)^2}{(a+b)(c+d)(a+c)(b+d)}$$

where a, b, c, d represent the frequencies in the four cells.

b) ***Cramer's V.*** Cramer's V describes the strength of association in a sample. Its value lies between 0.0 reflecting complete independence, and 1.0 showing complete dependence of the attributes.

$$V = \sqrt{\frac{\chi^2}{N(L-1)}}$$

where $L = \min(r, c)$.

c) ***Contingency coefficient.*** Like Cramer's V, the coefficient of contingency is used to describe the strength of association in a sample. Its upper limit is a function of the number of categories. The index cannot attain 1.0 .

$$\text{CC} = \sqrt{\frac{\chi^2}{\chi^2 + N}}$$

d) ***Degrees of freedom.***

$$\text{df} = (r-1)(c-1)$$

e) ***Adjusted N.*** This is the N used in the statistical computations, i.e. the number of cases with valid codes. It is weighted if a weight variable was specified.

f) ***S.*** S equals the number of agreements in order minus the number of disagreements in order. For a given cell in a table, all the cases in cells to the right and below are in agreement, all the cases to the left and below are in disagreement. S is the numerator of the tau statistics and of gamma.

$$S = \sum_{i=1}^{r-1} \sum_{j=1}^{c} f_{ij} \left(\sum_{h=i+1}^{r} \sum_{l=j+1}^{c} f_{hl} - \sum_{m=i+1}^{r} \sum_{n=1}^{j-1} f_{mn} \right)$$

where f_{ij}, f_{hl} and f_{mn} are the observed frequencies in cells ij, hl and mn respectively.

g) ***Variance of S.*** This is the variance of S when ties exist. (A tie is present in the data if more than one case appears in a given row or column.)

$$\begin{aligned}
\sigma_s^2 &= \frac{N(N-1)(2N+5) - \sum_j f_{\cdot j}(f_{\cdot j}-1)(2f_{\cdot j}+5) - \sum_i f_{i\cdot}(f_{i\cdot}-1)(2f_{i\cdot}+5)}{18} + \\
&+ \frac{\left[\sum_j f_{\cdot j}(f_{\cdot j}-1)(f_{\cdot j}-2)\right]\left[\sum_i f_{i\cdot}(f_{i\cdot}-1)(f_{i\cdot}-2)\right]}{9N(N-1)(N-2)} + \\
&+ \frac{\left[\sum_j f_{\cdot j}(f_{\cdot j}-1)\right]\left[\sum_i f_{i\cdot}(f_{i\cdot}-1)\right]}{2N(N-1)}
\end{aligned}$$

h) ***Standard deviation of S.***

$$\sigma_s = \sqrt{\sigma_s^2}$$

i) ***Normal deviation of S.*** It provides a large sample test of significance for tau or gamma with ties. The minus one in the numerator is a correction for continuity (if S is negative, unity is added). The value may be referred to a normal distribution table. The test is conditional to the distribution of ties.

$$Z = \frac{S-1}{\sigma_s}$$

j) ***Tau a.*** The Kendall's τ is a measure of association for ordinal data. Tau a assumes that there are no ties in the data, or that ties, if present, represent a "measurement failure" which is properly reflected by a reduced strength of relationship. Tau a can range from -1.0 to $+1.0$.

$$\tau_a = \frac{S}{\frac{N(N-1)}{2}}$$

k) ***Tau b.*** Tau b is like tau a except that ties are permitted, i.e. there may be more than one case in a given row or column of the bivariate table. Tau b can reach unity only when the number of rows equals the number of columns.

$$\tau_b = \frac{S}{\sqrt{\left[\frac{N(N-1)}{2} - T_1\right]\left[\frac{N(N-1)}{2} - T_2\right]}}$$

where

$$\begin{aligned} T_1 &= \left[\sum_i f_{i\cdot}(f_{i\cdot} - 1)\right] / 2 \\ T_2 &= \left[\sum_j f_{\cdot j}(f_{\cdot j} - 1)\right] / 2 \end{aligned}$$

l) ***Tau c.*** Tau c (also known as Kendall-Stuart tau) is like tau b except that if the number of rows is not equal to the number of columns, tau b cannot attain the values ± 1.0 while tau c can attain these values.

$$\tau_c = \frac{S}{1/2\, N^2\, [(L-1)/L]}$$

where $L = \min(r, c)$.

m) ***Gamma.*** The Goodman-Kruskal γ is another widely used measure of association that is closely related to Kendall's τ. It can range from -1.0 to $+1.0$ and can be computed even though ties occur in the data.

$$\gamma = \frac{S}{S_+ + S_-}$$

where

$$\begin{aligned} S &= S_+ - S_- \\ S_+ &= \text{the total number of pairs in like order} \\ S_- &= \text{the total number of pairs in unlike order.} \end{aligned}$$

n) ***Spearman's rho.*** This is an ordinary Pearson product moment correlation coefficient calculated on ranks. It ranges from -1.0 to $+1.0$. The Spearman's rho computed by TABLES incorporates a correction for ties.

The correction factor, T, for a single group of tied cases is:

$$T = \frac{t^3 - t}{12}$$

where t equals the number of cases tied at a given rank, i.e. the number of cases in a given row or a given column.

The Spearman's rho is calculated

$$\rho_s = \frac{\sum x^2 + \sum y^2 - \sum d^2}{2\sqrt{\sum x^2 \sum y^2}}$$

where

$$\begin{aligned}
\sum x^2 &= \frac{N^3 - N}{12} - \sum T_x \\
\sum y^2 &= \frac{N^3 - N}{12} - \sum T_y \\
\sum d^2 &= \sum_k (X_k - Y_k)^2 \\
\sum T_x &= \text{the sum of the } T\text{'s for all rows with more than 1 case} \\
\sum T_y &= \text{the sum of the } T\text{'s for all columns with more than 1 case} \\
X_k &= \text{the rank of case } k \text{ on the row variable} \\
Y_k &= \text{the rank of case } k \text{ on the column variable.}
\end{aligned}$$

Note that when more than one case occurs in a given row (or column), the value of the X_k's (or Y_k's) for the tied cases is the average of the ranks which would have been assigned if there had been no ties. For example, if there are 15 cases in the first row of a table, then those 15 cases would all be assigned a rank, i.e. X value, of 8.

o) ***Lambda symmetric.*** This lambda is a symmetric measure of the power to predict; it is appropriate when neither rows nor columns are specially designated as the thing predicted from, or known, first. Lambda has the range from 0 to 1.0 .

$$\lambda_{\text{sym}} = \frac{\sum_i \max_j f_{ij} + \sum_j \max_i f_{ij} - \max_j f_{\cdot j} - \max_i f_{i\cdot}}{2N - \max_j f_{\cdot j} - \max_i f_{i\cdot}}$$

where

$$\begin{aligned}
f_{ij} &= \text{the observed frequency in cell } ij \\
\max_j f_{ij} &= \text{the largest frequency in row } i \\
\max_i f_{ij} &= \text{the largest frequency in column } j \\
\max_j f_{\cdot j} &= \text{the largest marginal frequency among the columns } j \\
\max_i f_{i\cdot} &= \text{the largest marginal frequency among the rows } i.
\end{aligned}$$

p) ***Lambda A, row variable dependent.*** This lambda is appropriate when the row variable is the dependent variable. It is a measure of proportional reduction in the probability of error, when predicting the row variable, afforded by specifying the column category. The lambda row dependent has the range from 0 to 1.0 .

$$\lambda_{\text{rd}} = \frac{\sum_j \max_i f_{ij} - \max_i f_{i\cdot}}{N - \max_i f_{i\cdot}}$$

See above for the definition of the terms in this formula.

q) ***Lambda B, column variable dependent.*** This lambda is appropriate when the column variable is the dependent variable. It has the range from 0 to 1.0 .

$$\lambda_{\text{cd}} = \frac{\sum_i \max_j f_{ij} - \max_j f_{\cdot j}}{N - \max_j f_{\cdot j}}$$

See above for the definition of the terms in the formula.

r) ***Evidence Based Medicine (EBM) statistics.*** They are calculated for 2 x 2 tables where the first row represents frequences of event (a) and no event (b) for cases in the treated group, and the second row represents frequences of event (c) and no event (d) for cases in the control group.

The following statistics are calculated:

Experimental event rate

$$EER = a/(a+b)$$

Control event rate

$$CER = c/(c+d)$$

Absolute risk reduction (risk difference)

$$ARR = |CER - EER|$$

Relative risk reduction

$$RRR = ARR/CER$$

Number needed to treat

$$NNT = 1/ARR$$

Relative risk (risk ratio)

$$RR = EER/CER$$

and its 95% confidence interval

$$CI_{RR} = exp\left[\, ln(estimator\ RR) \pm 1.96\sqrt{T}\,\right]$$

where estimated variance of $ln(estimator\ RR)$ is

$$T = \frac{b/a}{a+b} + \frac{d/c}{c+d}$$

Relative odds (odds ratio)

$$OR = ad/bc$$

and its 95% confidence interval

$$CI_{OR} = exp\left[\, ln(estimator\ OR) \pm 1.96\sqrt{V}\,\right]$$

where estimated variance of $ln(estimator\ OR)$ is

$$V = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}$$

s) ***Fisher exact test.*** The Fisher exact probability test is an extremely useful non-parametric technique for analyzing discrete data (either nominal or ordinal) from two independent samples. It is used when all the cases from two independent random samples fall into one or the other of two mutually exclusive categories. The test determines whether the two groups differ in the proportion with which they fall into the two classifications.

Probability of observed outcome is calculated as follows:

$$p = \frac{(a+b)!\,(c+d)!\,(a+c)!\,(b+d)!}{N!\,a!\,b!\,c!\,d!}$$

where a, b, c, d represent the frequencies in the four cells.

The TABLES program gives also both one-tailed and two-tailed exact probabilities, called "probability of outcome equal to or more extreme than observed" and "probability of outcome as extreme as observed in either direction" respectively.

t) ***Mann-Whitney test.*** The Mann-Whitney U test can be used to test whether two independent groups have been drawn from the same population. It is a most useful alternative to the parametric t-test when the measurement is weaker than interval scaling. In the TABLES program it is required that the row variable be the dichotomous grouping variable.

Let

$$\begin{aligned} n_1 &= \text{the number of cases in the smaller of the two groups} \\ n_2 &= \text{the number of cases in the second group} \\ R_1 &= \text{sum of ranks assigned to group with } n_1 \text{ cases} \\ R_2 &= \text{sum of ranks assigned to group with } n_2 \text{ cases.} \end{aligned}$$

Then

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

and

$$U = \min(U_1, U_2)$$

If there are more than 10 cases in each group, the TABLES program provides Z approximation (normal approximation of U) calculated as follows:

$$Z = \frac{U - n_1 n_2/2}{\sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}}$$

u) ***Wilcoxon signed ranks test.*** The Wilcoxon test is a statistical test for two related samples and it utilizes information about both the direction and the relative magnitude of the differences within pairs of variables.

The sum of positive ranks, T^+, is obtained as follows:

- The signed differences $d_k = x_k - y_k$ are calculated for all cases.
- The differences d_k are ranked without respect to their signs. The cases with zero d_k's are dropped. The tied d_k's are assigned the average of the tied ranks.
- Each rank is affixed the sign (+ or −) of the d which it represents.
- N' is the number of non-zero d_k's.
- T^+ is the sum of positive d_k's.

If $N' > 15$, the program computes the Z approximation (normal approximation of T^+) as follows:

$$Z = \frac{T^+ - \mu_{T^+}}{\sigma_{T^+}}$$

where

$$\mu_{T^+} = \frac{N'(N'+1)}{4}$$

$$\sigma^2_{T^+} = \frac{N'(N'+1)\,(2N'+1)}{24} - \frac{1}{2}\sum_{t=1}^{g} n_t\,(n_t-1)\,(n_t-2)$$

and

$$\begin{aligned} g &= \text{the number of groupings of different tied ranks} \\ n_t &= \text{the number of tied ranks in grouping } t. \end{aligned}$$

Note that Z approximation is also adjusted for the tied ranks. The use of this, however, produces no change in variance when there are no ties.

v) ***t-test.*** This t-ratio is appropriate for testing the difference between two independent means, i.e. two independent samples. The variance is pooled.

$$t = \frac{\overline{y}_i - \overline{y}_h}{\sqrt{\left(\frac{n_i s_i^2 + n_h s_h^2}{n_i + n_h - 2}\right)\left(\frac{n_i + n_h}{n_i\, n_h}\right)}}$$

where

$$\begin{aligned}
\overline{y}_i &= \text{the mean of the column variable for cases in row } i \\
\overline{y}_h &= \text{the mean of the column variable for cases in row } h \\
s_i^2 &= \text{the sample variance of the column variable for cases in row } i \\
s_h^2 &= \text{the sample variance of the column variable for cases in row } h.
\end{aligned}$$

If t-tests are requested, sample standard deviations are calculated for the cases in each row as follows:

$$s_i = \sqrt{\frac{\sum y^2}{n_i} - \overline{y}_i^2}$$

57.3 Note on Weights

If bivariate statistics are requested and a weight variable is specified, a warning is printed and the statistics are computed using weighted values:

$$\begin{aligned}
x_k &= w_k\, x_k \\
x_k^2 &= w_k\, x_k^2 \\
y_k &= w_k\, y_k \\
y_k^2 &= w_k\, y_k^2 \\
N &= \sum_k w_k \\
f_{ij} &= \text{the weighted frequency in cell } ij.
\end{aligned}$$

Chapter 58

Typology and Ascending Classification

Notation

x	=	values of variables
k	=	subscript for case
v	=	subscript for variable
g, i, j	=	subscripts for groups
a	=	number of active variables (quantitative and dichotomized qualitative)
p	=	number of passive variables (quantitative and dichotomized qualitative)
t	=	number of initial groups
N_i	=	number of cases in group i (weighted if the case weight is used)
N_j	=	number of cases in group j (weighted if the case weight is used)
α	=	value of the variable weight
w	=	value of the case weight
W	=	total sum of case weights.

58.1 Types of Variables Used

The program accepts both QUANTITATIVE and QUALITATIVE (categorical) variables, the latter being treated as quantitative after full dichotomization of their respective categories, i.e. after the construction of as many dichotomic (1/0) variables as the number of categories. The variables used by the program may be either active or passive. The ACTIVE VARIABLES are those on the basis of which the typology is constructed. The PASSIVE VARIABLES do not participate in the construction of typology, but the program prints for them the main statistics within the groups of typology.

A set of active variables is denoted here $\mathcal{X}_a$, and a set of passive variables $\mathcal{X}_p$.

58.2 Case Profile

Profile of the case k is a vector $\mathcal{P}_k$ such as

$$\mathcal{P}_k = (x_{k1}, x_{k2}, \ldots, x_{kv}, \ldots, x_{ka}) = (x_{kv})$$

where all $x_v \in \mathcal{X}_a$.

If the active variables are requested to be standardized, the k^{th} case profile becomes

$$\mathcal{P}_k = \left(\frac{x_{kv}}{s_v}\right)$$

where s_v is the standard deviation of the variable x_v (see 7.b below).

58.3 Group profile

Profile of the group i, called also barycenter of group, is a vector $\mathcal{P}_i$ such as

$$\mathcal{P}_i = (\overline{x}_{i1}, \overline{x}_{i2}, \ldots, \overline{x}_{iv}, \ldots, \overline{x}_{ia}) = (\overline{x}_{iv})$$

and in the case of standardized data it becomes

$$\mathcal{P}_i = \left(\frac{\overline{x}_{iv}}{s_v}\right)$$

where the numerator is the mean of the variable x_v for the cases belonging to the group i and denominator is the overall standard deviation of this variable.

58.4 Distances Used

There are three basic types of distances used in the program, namely: city block distance, Euclidean distance and Chi-square distance of Benzécri. They may be used to calculate distances between two cases, between a case and a group of cases and between two groups of cases. Below, this distances are defined as distances between two groups of cases (between two group profiles), but the other distances can easily be obtained by adapting respective formulas.

a) ***City block distance.***

$$d_{ij} = d(\mathcal{P}_i, \mathcal{P}_j) = \frac{\sum_{v=1}^{a} \alpha_v \, |\overline{x}_{iv} - \overline{x}_{jv}|}{\sum_{v=1}^{a} \alpha_v}$$

b) ***Euclidean distance.***

$$d_{ij} = d(\mathcal{P}_i, \mathcal{P}_j) = \sqrt{\frac{\sum_{v=1}^{a} \alpha_v \, (\overline{x}_{iv} - \overline{x}_{jv})^2}{\sum_{v=1}^{a} \alpha_v}}$$

c) ***Chi-square distance.***

$$d_{ij} = d(\mathcal{P}_i, \mathcal{P}_j) = \sqrt{\sum_{v=1}^{a} \frac{1}{p_v} \left(\frac{p_{iv}}{p_i} - \frac{p_{jv}}{p_j}\right)^2}$$

where

$$p_v = \sum_{g=1}^{t} \overline{x}_{gv} \;, \quad p_i = \sum_{v=1}^{a} \overline{x}_{iv} \;, \quad p_j = \sum_{v=1}^{a} \overline{x}_{jv}$$

$$p_{iv} = \frac{\overline{x}_{iv}}{\sum_{g=1}^{t} \sum_{v=1}^{a} \overline{x}_{gv}} \;, \quad p_{jv} = \frac{\overline{x}_{jv}}{\sum_{g=1}^{t} \sum_{v=1}^{a} \overline{x}_{gv}}$$

Moreover, the program provides a possibility of using "weighted" distance, called DISPLACEMENT, which is defined as follows:

$$D_{ij} = D(\mathcal{P}_i, \mathcal{P}_j) = \frac{2N_iN_j}{N_i + N_j}\, d_{ij}$$

Note that displacement between two case profiles is equal to their distance since $N_i = N_j = 1$.

58.5 Building of an Initial Typology

a) ***Selection of an initial configuration.*** Before starting the process of aggregating the cases, the program selects the initial configuration, i.e. t initial group profiles, in either one of the following ways:

- case profiles of t randomly selected cases (using random numbers) constitute the starting configuration; in order to obtain the initial configuration, the remaining cases are distributed into t groups as described below;
- case profiles of t cases selected in a stepwise manner constitute the starting configuration; in order to obtain the initial configuration, the remaining cases are distributed into t groups as described below;
- the initial configuration is a set of group profiles calculated for cases distributed across categories of a key variable;
- the initial configuration is a set of "a priori" group profiles provided by the user.

When the construction starts from t case profiles, the program considers this set of t vectors as a set of t "starting cases" and distributes the remaining cases according to their distance to each of the starting case.

Let denote the set of t starting cases by

$$\mathcal{P}_{\text{starting}} = \Big\{\mathcal{P}_{k_1}, \mathcal{P}_{k_2}, \ldots, \mathcal{P}_{k_t}\Big\}$$

and the distance between groups and/or cases i and j by $D(\mathcal{P}_i, \mathcal{P}_j)$.

Note that $D(\mathcal{P}_i, \mathcal{P}_j)$ can be any distance defined in the section 4 above.

For each case $i \notin \mathcal{P}_{\text{starting}}$ the program calculates

$$\beta = \min_{1 \le j \le t} \Big[D(\mathcal{P}_i, \mathcal{P}_{k_j})\Big]$$

$$\gamma = \min\Big[D(\mathcal{P}_{k_1}, \mathcal{P}_{k_2}), D(\mathcal{P}_{k_1}, \mathcal{P}_{k_3}), \ldots, D(\mathcal{P}_{k_{t-1}}, \mathcal{P}_{k_t})\Big]$$

There are two possibilities:

- $\beta \le \gamma$: case i is assigned to the closest group $\mathcal{P}_{k_j}$ and the profile of this group is recalculated
$$\mathcal{P}_{k_j} = \Big(\mathcal{P}_{k_j} + \mathcal{P}_i\Big)/2$$
- $\beta > \gamma$: case i forms a new group which is added to the set $\mathcal{P}_{\text{starting}}$, and the two closest profiles $\mathcal{P}_{k_j}$ and $\mathcal{P}_{k_{j'}}$ are aggregated forming one group with the new profile
$$\mathcal{P}_{k_j} = \Big(\mathcal{P}_{k_j} + \mathcal{P}_{k_{j'}}\Big)/2$$

At the end of this procedure, the initial configuration is a set of t profiles

$$\mathcal{P}_{\text{initial}} = \Big\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_j, \ldots, \mathcal{P}_t\Big\}$$

where $\mathcal{P}_j$ is a mean profile of all the cases belonging to the group j.

At this stage the program does not take into account weighting of cases, if any.

b) ***Stabilization of the initial configuration.*** The initial configuration is stabilized by an iteration process. During each iteration, the program redistributes the cases among initial groups taking into account their distances to each group profile.

Here again there are two possibilities:

- when case $i \in \mathcal{P}_j$ and

$$D(\mathcal{P}_i, \mathcal{P}_j) = \min_{1 \le g \le t} \Big[D(\mathcal{P}_i, \mathcal{P}_g) \Big]$$

then this case remains in the group $\mathcal{P}_j$;

- when case $i \in \mathcal{P}_j$ but

$$D(\mathcal{P}_i, \mathcal{P}_{j'}) = \min_{1 \le g \le t} \Big[D(\mathcal{P}_i, \mathcal{P}_g) \Big]$$

then the case i is moved from the group $\mathcal{P}_j$ to the group $\mathcal{P}_{j'}$, and the profiles of those two groups are recalculated as follows:

$$\mathcal{P}_j = \left(N_j \mathcal{P}_j - \mathcal{P}_i \right) / (N_j - 1)$$

$$\mathcal{P}_{j'} = \left(N_{j'} \mathcal{P}_{j'} + \mathcal{P}_i \right) / (N_{j'} + 1)$$

After this operation, the group $\mathcal{P}_j$ contains $N_j - 1$ cases and the group $\mathcal{P}_{j'}$ contains $N_{j'} + 1$ cases.

Note that if the cases are weighted, then

$$N_j = N_j - w_i$$

$$N_{j'} = N_{j'} + w_i$$

$$\mathcal{P}_i = w_i \, \mathcal{P}_i$$

where w_i is the weight of the case i, and N_j and $N_{j'}$ are the weighted number of cases in the groups $\mathcal{P}_j$ and $\mathcal{P}_{j'}$ respectively.

Stability of groups is measured by the percentage of cases that do not change groups between two subsequent iterations.

The procedure is repeated until the groups are stabilized or when the number of iterations fixed by the user is reached.

58.6 Characteristics of Distances by Groups

a) ***N.*** The number of cases in each group of the initial typology.

b) ***Mean.*** Mean distance for each group, i.e. the mean of distances from the group profile over all cases belonging to this group.

c) ***SD.*** Standard deviation of distance for each group.

d) ***Classification of distances.*** Distribution of cases, both in terms of frequency and percentages, across 15 continuous intervals, which are different for each group.

e) ***Total count.*** Total number of cases participating in the building of the initial typology.

f) ***Mean.*** Overall mean distance.

g) ***SD.*** Overall standard deviation of distance.

h) ***Classification of distances (same limits for each group).*** Same as 6.d above except that the 15 intervals are of the same range for all groups.

58.7 Summary Statistics for Quantitative Variables and for Qualitative Active Variables

a) ***Mean.*** Mean of quantitative $x_v \in (\mathcal{X}_a \cup \mathcal{X}_p)$. For qualitative variable categories, it is a proportion of cases in this category.

$$\overline{x}_v = \frac{\sum_k w_k \, x_{kv}}{W}$$

b) ***S. D.*** Standard deviation.

$$s_v = \sqrt{\frac{W \sum_k w_k \, x_{kv}^2 - \left(\sum_k w_k \, x_{kv}\right)^2}{W^2}}$$

c) ***Weight.*** The value of variable weight calculated for each variable as follows:

$$\alpha_v = \begin{cases} 0 & \text{for quantitative passive variables} \\ 1 & \text{for quantitative active variables} \\ \frac{\sqrt{(c+1)/3}}{c} & \text{for categories of a qualitative active variable,} \\ & \text{where } c \text{ is the number of non-empty categories} \\ & \text{of the variable under consideration} \\ 1 & \text{for categories of a qualitative active variable} \\ & \text{if Chi-square distance is used.} \end{cases}$$

58.8 Description of Resulting Typology

At the end of the initial typology construction, and also at the end of each step of ascending classification, all variables, i.e. active and passive are evaluated by the amount of explained variance. It is a measure of discriminant power of each quantitative variable and each category of qualitative variables. This is followed by an individual description of all groups of the typology.

a) ***Proportion of cases.*** Percentage, multiplied by 1000, of cases belonging to each group of the typology.

b) ***Explained variance.***

$$\mathrm{EV}(x_v) = \frac{\sum_{i=1}^{t_g} N_i \left(\overline{x}_{iv} - \overline{x}_v\right)^2}{\sum_k w_k \left(x_{kv} - \overline{x}_v\right)^2} \times 1000$$

where

$$\begin{aligned} t_g &= \text{number of groups in the typology} \\ \overline{x}_{iv} &= \text{mean of the variable } v \text{ in group } i \\ \overline{x}_v &= \text{grand mean of the variable } v. \end{aligned}$$

c) ***Grand mean.***

For QUANTITATIVE variables, mean values as described under 7.a above.

For EACH CATEGORY OF QUALITATIVE variables, percentage of cases in this category.

d) ***Statistics for each group of the typology.***

For QUANTITATIVE variables:
first line: mean values as described under 7.a above;
second line: standard deviations as described under 7.b above.
For EACH CATEGORY OF QUALITATIVE variables:
first line: column percentage of cases;
second line: row percentage of cases.

58.9 Summary of the Amount of Variance Explained by the Typology

Similarly to the description of the resulting typology, a summary table is printed at the end of the initial typology construction and at the end of each step of ascending classification.

a) ***Variables explaining*** 80% ***of the variance.*** List of the most discriminating variables, i.e. those variables which – taken altogether – are responsible for at least 80% of the explained variance, together with the amount of variance explained by each of them individually (see 8.b above).

b) ***Mean variance explained by active variables.***

$$\overline{\mathrm{EV}}_{\mathrm{active}} = \frac{\sum_{v=1}^{a} \alpha_v \, \mathrm{EV}(x_v)}{\sum_{v=1}^{a} \alpha_v}$$

c) ***Mean variance explained by all variables.***

$$\overline{\mathrm{EV}}_{\mathrm{all}} = \frac{\sum_{v=1}^{a+p} \alpha_v \, \mathrm{EV}(x_v)}{\sum_{v=1}^{a+p} \alpha_v}$$

d) ***Mean variance explained by the variables which explain*** 80% ***of the total variance.*** After each regrouping, the program looks for variables which explain at least 80% of the total variance (see 9.a above) and prints mean variance explained by those variables before and after regrouping, and the percentage of such variables.

58.10 Hierarchical Ascending Classification

After creation of the initial typology, the program performs a sequence of regroupings, reducing one by one the initial number of groups up to the number specified by the user. At each regrouping, the program selects two closest groups, i.e. two groups with the smallest distance or displacement (see section 4 above), and calculates the profile for this new group.

a) ***Group i + j.*** Profile of the new group, printed for up to 15 active variables in descending order of their deviation (see 10.d below). Note that if there are less than 15 active variables, or less than 15 variables with valid cases in aggregated groups, the program completes the list using passive variables.

b) ***Group i.*** Profile of the group i, printed for the same variables as above.

c) ***Group j.*** Profile of the group j, printed for the same variables as above.

d) ***Dev.*** Absolute value of the difference between profiles of groups i and j, printed for the same variables as above.

$$\mathrm{Dev}(x_v) = |\overline{x}_{iv} - \overline{x}_{jv}|$$

e) ***Weighted deviation.*** Deviation weighted by the variable weight and the variable standard deviation, printed for the same variables as above.

$$\text{WDev}(x_v) = \text{Dev}(x_v)\ \frac{\alpha_v}{s_v}$$

58.11 References

Aimetti, J.P., *SYSTIT: Programme de classification automatique*, GSIE-CFRO, Paris, 1978.

Diday, E., *Optimisation en classification automatique*, RAIRO, Vol. 3, 1972.

Hall & Ball, A clustering technique for summarizing multivariate data, *Behavioral Sciences*, Vol. 12, No 2, 1967.

Appendix

Error Messages From IDAMS Programs

Overview

An effort has been made to make the error messages self-explanatory. Thus this Appendix essentially describes the coding scheme used for error messages.

Errors and Warnings

Errors (E) always cause termination of IDAMS program execution, while warnings (W) alert the user on possible abnormalities in the data and/or in the control statements, and also on possible misinterpretation of results. Error and warning messages have the following format:

`***E* aaannn` text of error message

`***W* aaannn` text of warning message

where

`nnn` is a three digit number, starting from 001 for warnings and from 101 for errors;

`aaa` indicates where the message comes from, according to the following rules:

- ***Messages from programs:*** the first letter of the program name followed by next two consonants in the program name.
- ***Messages from subroutines:***

  SYN general syntax errors;

  RCD Recode (syntax) errors and warnings;

  DTM data and dictionary errors, and warnings about data and dictionary files;

  SYS errors and warnings from the Monitor;

  FLM file management errors and warnings.

Fortran Run-Time Error Messages

When errors occur during program execution (run time) of a program, the Visual Fortran RTL issues diagnostic messages. They have the following format:

forrtl: severity (number): text

forrtl	Identifies the source as the Visual Fortran RTL.
severity	The severity levels are: severe (must be corrected), error (should be corrected), warning (should be investigated), or info (for informational purposes only).
number	This is the message number, also the IOSTAT value for I/O statements.
text	Explains the event that caused the message.

The run-time error messages are self-explanatory and thus they are not listed here.

Index

aggregation of data, 45, 50, 97
alphabetic variables, 13
analysis
  of correspondences, 193
  of time series, 311, 315
  of variance, 217, 231, 359, 371
analysis of variance
  multivariate, 225
auto-correlation, 315
auto-regression, 315

binary splits, 261, 389, 391, 392
bivariate
  statistics, 269, 294, 396
    output by TABLES, 272
  tables, 269, 293
    graphical presentation, 294
    output by TABLES, 272
blanks, 13
  detection, 112
  recoding, 29, 103
box and whisker plots, 307

C-records, 15
  listing, 143
  use in data validation, 109
case
  creating several cases from one, 49
  deletion, 127, 159
  identification (ID)
    correction, 127
  listing, 127, 143, 163
  principal, 193, 344
  selection
    with filter, 25
    with Recode, 49
  size limitations, 12
  specifying number of records per case, 14
  supplementary, 193, 346
categorical variables
  in regression, 201
checking
  codes, 58, 109
  consistency, 59, 115
  data structure, 58, 119
  range of values, 58, 109
  sort order, 159
chi-square
  distance, 285, 404
  test, 269, 294, 396
city block distance, 174, 215, 285, 320, 357, 404
classification of objects
  based on fuzzy logic, 172, 322
  based on hierarchical clustering, 172, 323, 324
  based on partitioning, 171, 320, 322
cluster analysis, 171, 319
code
  checking, 58, 109
  labels, 15
coefficients
  B, 203, 244, 257, 350, 378, 388
  beta, 203, 219, 350, 361
  constant term, 203, 244, 257, 350, 378, 388
  eta, 219, 232, 361, 372
  Gini, 189, 336
  multiple correlation, 203, 349
  of variation, 203, 219, 232, 269, 347, 359, 360, 371, 396
  partial correlation, 203, 348
  Pearson r, 243, 377
comments in IDAMS setup, 22
condition code
  checking between programs, 21
  setting for control statements errors, 21
configuration
  analysis, 177, 327
  centering, 327, 353
  matrix, 327, 353, 356
    input to CONFIG, 178
    input to MDSCAL, 214
    input to TYPOL, 284
    output by CONFIG, 178
    output by MDSCAL, 213
    output by TYPOL, 283
  normalization, 327, 353
  projection, 178
  rotation, 177, 327
  transformation, 177, 328
  varimax rotation, 178, 328
consistency checking, 59, 115
contingency
  coefficient, 269, 294, 397
  tables, 269
continuation line
  control statements, 25
  Recode statements, 33
control statements, 24
  filter, 25
  label, 26
  parameters, 27
  rules for coding, 25

copying
datasets, 159
correcting
case ID, 127
data, 58, 88, 127
dictionary, 86
variables, 127
correlation
analysis, 243, 377
coefficients, 243, 377
matrix, 341, 348, 378
input to CLUSFIND, 172
input to MDSCAL, 213
input to REGRESSN, 204
output by PEARSON, 244
output by REGRESSN, 202, 203
partial, 203, 348
correspondence analysis, 193
covariance matrix, 341, 378
output by PEARSON, 245
Cramer's V, 269, 294, 397
cross-spectrum, 316
crosstabulations, 269

data
aggregation, 97
correction, 58, 88, 127
editing, 14, 57, 103
entry, 88
export
in DIF format, 134
in free format, 90, 134
format in IDAMS, 12
import, 19
in DIF format, 135
in free format, 89, 135
in the input stream, 22
listing, 143
recoding, 59
sorting, 88
structure checking, 58, 119
transformation, 59, 163
validation, 57, 109, 115, 119
dataset
building, 103
copying, 159
definition in IDAMS, 11
merging, 147
subsetting, 159
ddname, 23
for dictionary and data files, 30
deciles, 189, 271, 335, 396
decimal places, specification, 15
defaults in IDAMS parameters, 27
deleting
cases, 127, 159, 163
variables, 159, 163
densities, 305
descriptive statistics, 97, 98, 194, 257, 269, 291, 292, 339, 387, 395
dictionary, 14
code label (C-records), 15
copying, 159
creation, 86, 103
descriptor record, 14
example, 16
in the input stream, 22
listing, 143
variable descriptor (T-record), 14
verification, 86
discriminant
analysis, 183, 331
factor analysis, 184, 333
function, 183, 332
distance
chi-square, 285, 404
city block, 174, 215, 285, 320, 357, 404
Euclidean, 174, 211, 215, 285, 320, 356, 404
Mahalanobis, 183, 332
distribution
frequencies, 269
function, 189, 335
dummy variables
creation with Recode, 46
used in regression, 201
duplicate
cases, deletion, 159, 161
records, detection and deletion, 120
Durbin-Watson (test), 203, 351

EBM statistics, 269, 400
editing
data, 57
non-numeric data values, 29, 103
text files, 93
eigenvalues, 341
eigenvectors, 341
ELECTRE ranking method, 249
error messages, 411
Euclidean distance, 174, 211, 215, 285, 320, 356, 404
export
of data, 90, 133
of datasets, 6
of matrices, 6, 133
of multidimensional tables, 294

F-test, 203, 219, 232, 349, 372
factor analysis, 184, 193, 333, 339
files
data file, 79
dictionary file, 79
matrix file, 79
merging, 147, 155
names, 79
results file, 79
setup file, 79
size limitations for IDAMS, 12
sorting, 155
specifying in IDAMS, 22
system files, 80

permanent, 80
temporary, 80
used in WinIDAMS, 79
user files, 79
filter
control statement, 25
local
in ONEWAY, 234
in QUANTILE, 192
in SCAT, 260
in TABLES, 274
placement, 25
rules for coding, 25
syntax verification, 91
with R-variables, 49
Fisher
exact test, 269, 400
F-test, 203, 219, 232, 349, 372
folders
default folders, 80
used in WinIDAMS, 80
frequency distributions, 269, 291
frequency filters, 316
fuzzy logic
classification of objects, 172, 322
ranking of alternatives, 249, 384, 385

gamma (statistic), 269, 294, 398
Gini (coefficient), 189, 336
graphical exploration of data, 301
grouping data cases, 97

hierarchical clustering
agglomerative, 172, 323
based on dichotomic variables, 172, 324
divisive, 172, 324
histograms, 305, 315

IDAMS
control statements, 24
dataset, 11
building, 103
dictionary, 14
error messages, 411
execution of programs, 92
matrix, 16
export, 133
import, 133
results handling, 92
setup, 21
preparation, 90
verification, 91
IDAMS commands, 21
$CHECK, 21
$COMMENT, 22
$DATA, 22
$DICT, 22
$FILES, 22
$MATRIX, 22
$PRINT, 22
$RECODE, 22
$RUN, 22
$SETUP, 23
import
of data, 133
of data files, 89
of datasets, 6
of matrices, 6, 133
interaction
definition, 217
detection and treatment, 217
inverse matrix, 203, 348

Kaiser criterion, 197
Kendall's taus, 269, 294, 398
keywords
for common parameters, 29
rules for coding, 28
types, 27
Kolmogorov-Smirnov (D test), 189, 192, 336
kurtosis, 340, 396

label
control statement, 26
for code categories, 15
for variables, 15
placement, 26
rules for coding, 27
lambda statistics, 269, 294, 399
listing
cases, 127, 143
data, 143, 163
dictionary, 143
Lorenz
curve, 336
function, 189, 336

Mahalanobis distance, 183, 332
Mann-Whitney (test), 269, 401
marginal distributions, 269
matrix
export (free format), 134
import (free format), 135
in the input stream, 22
inverse, 203, 348
of correlations, 341, 348, 378
input to CLUSFIND, 172
input to MDSCAL, 213
input to REGRESSN, 204
output by PEARSON, 244
output by REGRESSN, 202, 203
of covariances, 341, 378
output by PEARSON, 245
of cross-products, 203, 244, 347, 348, 378
of dissimilarities, 171, 320
input to CLUSFIND, 172
input to MDSCAL, 213
of distances, 178, 328
output by CONFIG, 178
of partial correlations, 203, 348

of relations, 193, 194, 249, 340, 382, 383
of scalar products, 178, 328, 341
of similarities
input to CLUSFIND, 172
input to MDSCAL, 213
of statistics, 269
output by TABLES, 272
of sums of squares, 203, 347, 348
projection, 308
rectangular, 18
square, 16
vector of means and SD's, 18
mean, 319, 331, 339, 347, 359, 360, 365, 371, 377, 378, 387, 395, 407
merging
datasets, 147
at different levels, 147
at the same level, 147
files, 155
Minkowski r-metric, 211, 356
missing data
case-wise deletion
in PEARSON, 243
in REGRESSN, 202
checking for with Recode, 45
codes
assignment by Recode, 50
specification, 13, 15
definition, 13
handling by Recode, 34
pair-wise deletion
in PEARSON, 243
to be used for checking, 30
multidimensional scaling, 211, 353
multidimensional tables, 293
multiple classification analysis, 217
multivariate analysis of variance, 225

n-tiles, 189, 271, 335, 396
non-numeric data values, 13
detection, 103
editing, 29, 103
non-parametric tests
Fisher (exact), 269, 400
Mann-Whitney, 269, 401
Wilcoxon (signed ranks), 269, 401
normalization
of configuration, 327, 353
of relation matrix, 249, 384
numeric variables, 103
coding rules, 12

outliers
definition, 222, 264
detection and elimination, 222
identification and printing, 262

parameters
common
BADDATA, 29
INFILE, 30
MAXCASES, 30
MDVALUES, 30
OUTFILE, 30
VARS, 30
WEIGHT, 30
default values, 27
parameter statements, 27
placement, 27
presentation in the Manual, 27
rules for coding, 28
types of keyword, 27
partial
correlation coefficients, 203, 348
order scoring, 235, 373
partitioning around medoids, 171, 320, 322
Pearson (correlation coefficient r), 243, 377, 388
Phi (statistic), 294
plotting scattergrams, 257
preference
data
example, 251
types of, 249, 379
strict, 250
weak, 250
principal components factor analysis, 193
printing IDAMS setup, 22

quantiles, 189, 271, 335, 396

random values
generation by Recode, 41
ranking analysis, 249, 379
classical logic, 249, 380
fuzzy logic, 249, 384, 385
Recode
accessing the Recode facility, 22
arithmetic functions, 36
constants
character, 35
numeric, 35
continuation line, 33
elements of language, 35
expressions, 36
arithmetic, 36
logical, 36
format of statements, 33
initialization of variable values, 34
logical functions, 44
missing data handling, 34
operands, 35
operators
arithmetic, 35
logical, 36
relational, 36
restrictions, 54
statements, 45
syntax verification, 91
testing, 34
V- and R-variables, 35

Recode, arithmetic functions
ABS, 37
BRAC, 37
COMBINE, 38
COUNT, 39
LOG, 39
MAX, 39
MD1, MD2, 40
MEAN, 40
MIN, 40
NMISS, 40
NVALID, 41
RAND, 41
RECODE, 41
SELECT, 42
SQRT, 42
STD, 43
SUM, 43
TABLE, 43
TRUNC, 44
VAR, 44
Recode, logical functions
EOF, 45
INLIST, 45
MDATA, 45
Recode, statements
assignment, 45
BRANCH, 48
CARRY, 50
CONTINUE, 48
DUMMY, 46
ENDFILE, 48
ERROR, 48
GO TO, 48
IF, 49
MDCODES, 50
NAME, 51
REJECT, 49
RELEASE, 49
RETURN, 49
SELECT, 47
recoding data, 31, 33, 59
example, 33, 51, 60
saving recoded variables, 163
record
duplicate record detection and deletion, 120
invalid record deletion, 119
missing record detection and padding, 120
regression, 201, 244, 257, 347, 378, 388
descending stepwise, 201, 352
lines, 306
multiple linear, 201, 347
stepwise, 201, 351
with categorical variables, 201, 206, 217
with dummy variables, 201, 206
with zero intercept, 352
repetition factor
in TABLES, 274
residuals, 351, 362, 391–393
output by MCA, 217, 219
output by REGRESSN, 202, 204
output by SEARCH, 261, 262
rotation of configuration, 177, 327

saving recoded variables, 163
scaling analysis, 211, 353
scatter plots, 257
3-dimensional, 308
grouped plot, 307
manipulation, 304
rotation, 308
scores
calculated by FACTOR, 194, 345, 346
calculated by POSCOR, 236, 375
scoring analysis, 235, 373
segmentation analysis, 261, 389
selecting cases with filter, 25
skewness, 340, 396
Sormer's D, 294
sort order checking, 129, 159
sorting files, 88, 155
spatial analysis, 177, 327
Spearman's rho, 269, 398
spectrum, 315
standard deviation, 331, 339, 347, 359, 360, 371, 377, 378, 387, 388, 396, 407
standardization
of measurements, 171, 319
of variables, 404
Student (t-test), 269, 402
subset specifications
in POSCOR, 239
in QUANTILE, 191
in TABLES, 274
subsetting
cases, 25
datasets, 159

T-records, 14
t-tests of means, 269, 402
tau statistics, 269, 294, 398
test
chi-square, 269, 294, 396
D of Kolmogorov-Smirnov, 189, 192, 336
Durbin-Watson, 203, 351
Fisher (exact), 269, 400
Fisher F, 203, 219, 232, 349, 372
Mann-Whitney, 269, 401
t of Student, 269, 402
Wilcoxon (signed ranks), 269, 401
testing
program control statements, 30
recode statements, 34
time series
analysis, 311
transformation, 314
transformation
of configuration, 177, 328
of data, 59, 163

of time series, 314
trend estimation, 315

univariate
statistics, 97, 98, 194, 203, 257, 269, 291, 292, 305, 315, 339, 387, 395
tables, 269, 293
graphical presentation, 294
output by TABLES, 272

validation of data, 57, 109
variable
active, 281, 403
aggregated, 97, 98
alphabetic, 13
correction, 127
decimal, 12
descriptor record, 14
dummy, 46
name, 15, 51
number, 12, 15
numeric, 12
coding rules, 12
editing, 14, 103, 105
passive, 281, 403
principal, 193, 342
reference number, 15
supplementary, 193, 343
type, 15
variable list
rules for coding, 30
variance analysis, 231, 371
varimax rotation
of configuration, 178, 328
of factors, 194, 346

weighting data, 30
Wilcoxon (signed ranks test), 269, 401
WinIDAMS
files, 79
folders, 80
User Interface
customization of environment, 83